**СОДЕРЖАНИЕ**

|  |  |  |
| --- | --- | --- |
|  | ВВЕДЕНИЕ | 4 |
| 1 | ТЕОРЕТИЧЕСКАЯ ЧАСТЬ | 7 |
| 1.1 | Обоснование потребности в web-сайте | 7 |
| 1.2 | Понятие web-сайта | 7 |
| 1.3 | Классификация web-сайтов | 8 |
| 1.4 | Проектирование web-сайта | 15 |
| 1.5 | Этапы разработки web-сайта | 15 |
| 1.6 | Постановка задачи при проектировании web-сайта | 17 |
| 1.7 | Выбор программного средства для разработки | 18 |
| 1.8 | Разработка интерфейса | 22 |
| 1.9 | Разработка базы данных | 23 |
| 1.10 | Разработка административной части сайта | 24 |
| 1.11 | Обоснование выбора среды разработки | 25 |
| 2 | ПРОЕКТНАЯ ЧАСТЬ | 26 |
| 2.1. | Общая структура сайта | 26 |
| 2.2 | Разработка интерфейса web-сайта | 26 |
| 2.3 | Дизайн web-сайта | 27 |
| 2.4 | Описание создания страниц web-сайта | 33 |
| 3 | ЭКОНОМИЧЕСКАЯ ЧАСТЬ | 35 |
| 3.1 | Расчет времени на создание программного продукта | 35 |
| 3.2 | Расчет себестоимости 1-го машино-часа работы компьютере | 40 |
| 3.3 | Расчёт расходов на содержание и эксплуатацию ПК | 43 |
| 3.4 | Расчет себестоимости программного продукта | 43 |
| 3.5 | Расчет срока окупаемости | 44 |
| 4 | МЕРОПРИЯТИЯ ПО ТЕХНИКЕ БЕЗОПАСНОСТИ И ОХРАНЕ ОКРУЖАЮЩЕЙ СРЕДЫ | 46 |
| 4.1 | Общие принципы и законодательство по охране труда | 46 |
| 4.2 | Пожарная безопасность и электро-безопасность | 46 |
| 4.3 | Санитарно-гигиенические требования к помещениям | 50 |
| 4.3.1 | Требования к освещению помещений и рабочих мест | 50 |
| 4.3.2 | Требования к организации и оборудованию рабочих мест | 51 |
| 4.3.3 | Требования к шуму и вибрации в помещениях | 53 |
| 4.4 | Режим труда и отдыха при работе с компьютером | 53 |
| 4.5 | Медико-профилактические и оздоровительные мероприятия | 55 |
|  | ЗАКЛЮЧЕНИЕ | 56 |
|  | СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ | 58 |
|  | ПРИЛОЖЕНИЕ А – ЛИСТИНГ ПРОГРАММЫ | 59 |
|  | ПРИЛОЖЕНИЕ Б – СОДЕРЖАНИЕ ДИСКА | 67 |

**ВВЕДЕНИЕ**

Система образования – одна из самых значимых отраслей в современном обществе. От качества подготовки специалистов напрямую зависит будущее как населения одной страны, так и всего мира в целом. Поэтому повышение качества образования – то, чему стоит уделять особое внимание. В настоящее время постоянно внедряются новые технологии, направленные на оптимизацию существующих систем, перевод услуг в электронный формат.

Учебный процесс колледжа – сложная система, состоящая из взамосвязанных процессов по планированию, составлению расписания, проведению занятий, аттестации и т.д. Ее участниками помимо преподавателей и студентов также являются заместители руководителя, заведующие отделениями, диспетчера, административный персонал.

Исходными данными, отправной точкой является учебный план. На его основе составляются расписание занятий, тематические планы, журналы, график итоговой аттестации. Все это требует большой внимательности и ответственности, так как малейшая ошибка может привести к неправильному графику проведения или срыву занятий, а также сказаться на заработной плате преподавателей. Преподаватели и заведующие тратят большое количество времени на сверку учебных планов, помимо своей основной работы. В связи с этим возникает необходимость автоматизировать этот процесс, чтобы снизить вероятность ошибок из-за человеческого фактора.

Исходя из вышеизложенного, тема дипломного проекта: «Разработка автоматизированной информационной системы учебного процесса колледжа», целью которого является: изучение, проектирование и разработка сайта, предназначенного для автоматизации работы с учебными планами, расписанием и журналами.

**В данном проекте была поставлена задача изучить основные механизмы учебного процесса, функции сотрудников, последовательность выполнения задач и на основе полученных данных спроектировать и разработать веб-сервис. Данная система реализует такие функции как загрузка данных о контингенте, персонале и учебных планах из файла, генерация выходных документов, автоматизированное составление расписания, учет вычитки часов, аттестация студентов, статистика и форум.**

Объектом исследования является деятельность коммунального государственного предприятия «Колледж информационных технологий».

В соответствии с целью исследования сформулированы следующие задачи:

* определить виды деятельности, которые целесообразно автоматизировать;
* выявить функциональные возможности веб-сайтов и их применение;
* разработать веб-сайт.

Актуальность дипломного проекта заключается в том, чтобы облегчить работу преподавателей и заведующих, снизить вероятность ошибок, сократить время, затрачиваемое на рутинную работу, а также создать возможность для студентов отслеживать свою динамику онлайн и дистанционно общаться друг с другом и с преподавателями на форуме.

Теоретико-методологическую базу данного исследования составляют учебная и методическая литература, ресурсы в сети Интернет.

Практическая значимость: при соблюдении принципа дноразовости ввода данных все модули системы связаны между собой, что исключает необходимость постоянной ручной сверки и выполнения двойной работы. Также данная система может помочь в организации удаленного обучения.

Для разработки системы были использованы языки программирования PHP, JavaScript, язык гипертекствой разметки HTML и каскадная таблица стилей CSS.

**1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ**

**1.1 Обоснование потребности в web-сайте**

В последние десятилетия все больше областей человеческой деятельности становятся автоматизированными. Система образования не исключение. Цель автоматизации – упрощение работы сотрудников и удобство пользователей. Наиболее распространенной и оптимальной формой автоматизации является веб-сайт, так как для работы с ним не требуется что-либо устанавливать и можно получить доступ к нему в любое время с любого устройства, имеющего выход в Интернет.

В соответствии с целью исследования сформулированы следующие задачи:

* изучить функции деятельности учреждения, которые необходимо автоматизировать;
* спроектировать структуру базы данных и сайта;
* выявить функциональные возможности веб-сайтов и их применение;
* разработать веб-сайт.

**1.2 Понятие web-сайта**

Информация, доступная пользователям Internet, располагается на Web-серверах, на которых установлено специальное программное обеспечение. Значительная часть этой информации организована в виде Web-сайтов. Каждый из них имеет свое имя (адрес) в Internet.

Web-сайт – это информация, представленная в определенном виде, которая располагается на Web-сервере и имеет свое имя (адрес). Для просмотра Web-сайтов на компьютере пользователя используются специальные программы, которые называются браузерами. В зависимости от того, какое имя (адрес) сайта мы зададим в строке «Адрес», браузер будет загружать в свое окно соответствующую информацию.

Web-сайт состоит из связанных между собой Web-страниц. Web-страница представляет собой текстовый файл с расширением «\*.htm», который содержит текстовую информацию и специальные команды – HTML-коды, определяющие в каком виде эта информация будет отображаться в окне браузера. Вся графическая, аудио- и видео-информация непосредственно в Web-страницу не входит и представляет собой отдельные файлы с расширениями «\*.gif», «\*.jpg» (графика), «\*.mid», «\*.mp3» (звук), «\*.avi» (видео). В HTML-коде страницы содержатся только указания на такие файлы.

Каждая страница Web-сайта также имеет свой Internet адрес, который состоит из адреса сайта и имени файла, соответствующего данной странице. Таким образом, Web-сайт – это информационный ресурс, состоящий из связанных между собой гипертекстовых документов (Web-страниц), размещенный на Web-сервере и имеющий индивидуальный адрес. Посмотреть Web-сайт может любой человек, имеющий компьютер, подключенный к Internet.

**1.3 Классификация web-сайтов**

В настоящее время в сети Интернет (глобальной, всемирной компьютерной сети) существует огромное количество сайтов. Эти сайты самые разные и отличаются друг от друга по очень большому числу параметров.

Изначально единственной технологией, с помощью которой создавались сайты и веб-страницы, была технология **«html».** Однако такие страницы и сайты выглядели бедно и не обладали**интерактивностью,** поэтому за короткое время, прошедшее от начала массового распространения интернета, появились и стали использоваться при создании сайтов многочисленные дополнительные технологии и языки.

Итак, классифицируем сайты по используемым технологиям:

**1) статические сайты и веб-страницы;** Статические сайты и веб-страницы — это сайты, сделанные по классической технологии «html». Веб-страницы таких сайтов написаны полностью на языке «html» и имеют расширение «\*.htm» или «\*.html». Статическими они называются потому, что, набирая определённый адрес (URL), соответствующий сайту или его определённой странице, вы всегда будете видеть одно и то же содержание. Это потому, что статические html-страницы лежат на сервере в неизменном виде и по вашему запросу сервер просто берёт их и отправляет вам в браузер. С динамическими страницами, которые не существуют в заранее предопределённом виде на сервере, а формируются при помощи серверных скриптов, ситуация иная, и, набирая один и тот же адрес, вы, в зависимости от разных условий, можете получать совершенно разное содержание.

Статические сайты имеют свои как плюсы, так и минусы. Плюсы статических сайтов — это простота их создания и нетребовательность к аппаратным ресурсам. Статические сайты не требуют поддержки скриптов и баз данных и могут быть размещены на абсолютно любом хостинге, при этом они не будут создавать почти никакой нагрузки на сервер, и с ними не возникнет проблема, когда сайт тормозит или выдаёт различные сбои при большом притоке посетителей. Статические сайты могут включать в себя графику, анимацию и **javascript**, они прекрасно подходят для предоставления пользователям информации, которая не должна, по крайней мере, часто и существенно изменяться. Основным минусом статических сайтов является отсутствие интерактивности. Кроме того, в таких сайтах сложнее изменить информацию, внешний вид и дизайн страниц.

**2) динамические сайты и веб-страницы;** Как уже было упомянуто, динамические страницы не существуют в неизменном виде на сервере, а формируются при помощи скриптов. Содержание и вид динамической страницы с одним и тем же адресом может быть совершенно различным в зависимости от разных условий — например, в зависимости от времени, от конкретного пользователя, от введённого пользователем запроса и т. п. Скрипты, которые используются для формирования динамических веб-страниц, могут быть написаны на разных языках. Распространёнными языками веб-программирования являются, например**, php, perl, asp** и т. д. Такие сайты могут предоставлять пользователям **интерактивность —** например, вводить разные запросы и осуществлять поиск по сайту, отправлять на сайт и сохранять свою собственную информацию и осуществлять общение с другими пользователями (как, например, в гостевых книгах, форумах и т. п.) и многие другие функциональные возможности. Кроме того, ведение и обновление этих сайтов гораздо проще. Минусы динамических сайтов в том, что они требуют поддержку дополнительных технологий, создают повышенную нагрузку на сервер, их сложнее оптимизировать, а ещё использование на сайте скриптов означает потенциальную угрозу для безопасности.

**3) флэш-сайты;** Технология флэш **(flash)** позволяет создавать очень красивые, интерактивные, со звуком и анимацией, сайты, которые выглядят на порядок эффектнее, чем обычные **html**-сайты. Однако сложность и трудоёмкость их изготовления, а также то, что страницы, созданные на флэш, имеют, как правило, большой вес и долго загружаются, что отпугивает часть пользователей, приводит к тому, что флэш-сайты не получили пока слишком широкого распространения.

Также достаточно просто построить классификацию сайтов, исходя из того, чьим является сайт, кто его владелец и создатель. Таким образом, вторая классификация сайтов:

Сейчас в интернете создать свой сайт может практически каждый. Только доменов второго уровня там уже многие миллионы, не говоря даже о сайтах, имеющих домены третьего уровня, которые во множестве расположены на бесплатных хостингах, таких, как narod.ru.

По принадлежности сайты подразделяются на:

**1) личные (персональные) сайты;**таким сайтом владеет и осуществляет его поддержку один человек. Такие сайты сейчас составляют большинство. Спектр этих сайтов весьма широк — от маленьких домашних страничек (home page), где есть немного информации о владельце и его увлечениях, до очень известных и популярных ресурсов.

**2) сайты коммерческих организаций; э**тих сайтов тоже сейчас очень много. По мере того, как интернет набирает популярность (а в западных странах к нему приобщены уже более половины населения), всё больше фирм и компаний рассматривают интернет как удобную среду если не для прямого ведения бизнеса и продажи товаров в онлайн-магазинах, то, по крайней мере, для рекламы своих товаров и услуг. По степени развитости присутствия коммерческой организации в интернете среди коммерческих сайтов можно выделить несколько подвидов: **сайты-визитки**, содержащие лишь небольшую общую информацию о фирме, такие сайты обычно не обновляются; т. н. **промо-сайты** или **сайты для бизнеса**, которые предназначены для презентации и продвижения различных товаров и услуг. На таких сайтах обычно оставляют контактные данные — адреса, телефоны, схемы проезда, предназначенные для потенциальных покупателей; и, наконец, полноценные **сайты электронной коммерции** или **интернет-магазины**, на которых можно заказать услуги или купить товар, не вставая с кресла перед монитором.

**3) сайты некоммерческих организаций;** разнообразные некоммерческие организации также стремятся заявить о своём существовании в сети интернет. Многообразие подобных сайтов огромно. Тут и сайты какого-нибудь регионального отделения пенсионного фонда, и сайты политических партий и общественных движений, и сайты научно-исследовательских институтов и т. д. и т. п. Весьма популярны в интернете новостные ресурсы, ресурсы, представляющие онлайновые версии различных СМИ. Для двух особых категорий сайтов некоммерческих организаций, а именно сайтов государственных и образовательных учреждений в интернете даже выделены специальные доменные зоны: **«\*.gov»** и **«\*.edu»**. Причём, в отличие от популярных доменных зон **«\*.com»**, **«\*.org»**, национальных доменных зон и т. п., попасть туда может далеко не каждый желающий. Таким образом, расположение сайта в зоне **«.edu»**, например, — это гарантия солидности образовательного учреждения.

Ещё одной достаточно формальной классификацией будет классификация сайтов по их масштабу, по уровню возложенных на них функций. Итак, третья классификация:

По величине и уровню решаемых задач сайты могут отличаться очень сильно. Есть сайты, состоящие из одной-единственной странички, а есть сайты, содержащие так много информации и требующие так много ресурсов, что для их поддержания используется с десяток отдельных высокопроизводительных серверов.

Приведём условную классификацию сайтов по этим параметрам:

**1) простые сайты, содержащие немного информации и состоящие из нескольких страничек** («сайты-визитки», домашние странички и т. п.). Основная цель их создателей и владельцев, как правило, — **просто обозначить своё присутствие в интернете**. Такие сайты обычно содержат лишь немного информации о владельце и совсем мало или вовсе не содержат какой-либо другой интересной пользователям сети информации.

**2) тематические, узконаправленные сайты.** Такие сайты могут быть довольно большими и подробно освещать какой-либо вопрос, предоставляя пользователям интернета, интересующимся определённой темой, возможность получить по ней хорошую и детальную информацию. Тематический сайт может рассказывать, например, об истории Древнего Рима, об отдыхе на Канарских островах, о выращивании и разведении кроликов и т. п. Многие из тематических сайтов занимают достойное место в интернете. Однако тематические, узконаправленные сайты могут привлекать и быть интересны лишь определённой категории пользователей.

**3) многофункциональные сайты (порталы).** Особое место в ряду интернет-сайтов занимают монстры интернета — многофункциональные и многотематические сайты-порталы. Создавать, поддерживать и обеспечивать работу таких сайтов сложно, но зато такие сайты могут привлекать практически все категории пользователей и каждому быть интересны. Сайты-порталы содержат в себе массу самой разнообразной информации и функций, на них можно узнать и свежие новости, и прогноз погоды, и прочитать гороскоп, и завести почтовый ящик, и вести блог, и создать сообщество по интересам и т. д. и т. п. Такие крупные порталы, как, например, Яндекс или Мэйл.ру известны практически каждому пользователю Рунета.

Следующая классификация будет, наверно, самой разнообразной и с ней будет труднее всего разобраться. Большинство сайтов предназначено для обычных пользователей сети, и спектр целей, с которыми тот или иной пользователь может придти в интернет, очень широк. Это могут быть и поиск информации, причём по самым различным темам и самого различного свойства, и общение, и использование онлайн-сервисов и т .п. Для достижения этих целей пользователь должен найти подходящий сайт (или сайты), а владелец или владельцы сайта должны каким-то образом привлечь на свой сайт целевую аудиторию, т. е. тех, кому этот сайт был бы интересен. Итак, четвёртая классификация — это попытка выделить.

**Виды сайтов по типу информации и возможностей, которые они предоставляют пользователям сети.**

Более-менее детальная классификация сайтов по типу информации и возможностям, предоставляемым пользователям интернета, очень обширна. Построить такую классификацию по этим признакам, которая точно распределила все сайты по подходящим для них категориям, довольно проблематично. Крупные каталоги интернет-ресурсов, такие как **Яндекс-каталог**, **DMOZ**, **list.mail.ru** и т. д., пытаются решать эту задачу по-разному. Множество аспектов и соображений, исходя из которых, можно выделять категории и подкатегории, оставляют много места для произвола.

Тем не менее, постараемся и здесь выделить типы и дать некую классификацию.

Вначале выделим наиболее общие группы сайтов **по назначению**, соотнеся их с максимально общими целями, ради которых пользователи приходят в интернет. Изначально интернет был средой для получения информации. И сейчас сайты, направленные на предоставление разнообразной информации пользователям, являются наиболее многочисленной группой. Итак, первая группа — **сайты, существующие для предоставления пользователям информации, контента**. Помимо собственно поиска информации, люди часто приходят в интернет ради общения, ради поиска людей, близких по интересам и т. п. Таким образом, вторая группа — **сайты, направленные на организацию общения и взаимодействия между пользователями**. Вскоре после появления интернета в него пришёл бизнес, и появилась третья группа сайтов. Это — сайты, предназначенные для предоставления услуг, продажи товаров через интернет, **сайты, основной функцией которых является организация электронной коммерции**. И, наконец, ещё одной важной группой сайтов, являются **сайты, предоставлящие различные онлайн-сервисы**.

Напишем теперь об этих группах сайтов поподробнее:

**1) сайты, предоставляющие контент**. В этой группе, в свою очередь, можно выделить множество разных типов сайтов по разным признакам. Во-первых, **сам вид контента может быть различным** — т. е. это может быть как текст, так и фотографии, рисунки, всякие звуковые файлы, например, музыка популярных исполнителей, и видео. По х**арактеру предоставляемого контента** можно выделить сайты информационно-тематические, новостные, развлекательные, сайты-библиотеки, сайты-базы определённого рода документов, например, база рефератов, разнообразные сайты-справочники, онлайн-энциклопедии и словари, сайты-каталоги, обобщающие информацию о других сайтах и т. д. Ну и, конечно же, **по тематике**. Однако список возможных тематик был бы настолько большим, что мы не будем приводить его на этом сайте. Хорошая классификация различных сайтов по тематикам представлена в Яндекс-каталоге.

**2) сайты для онлайн-контактов и общения**. Список типов сайтов в этой группе также очень велик. Сюда можно отнести и «классические» формы организации общения, такие как форумы, чаты, доски объявлений, так и новые, которые можно условно отнести к группе т. н. «веба 2.0″. Среди эти новых — популярные в последнее время «социальные сети», всякие блоги-сообщества в одном флаконе, «вопрос-ответные» проекты и т. п. Сюда нужно отнести и такие сайты, как службы знакомств, сайты, организующие общение между людьми, ищущими в сети работу и работодателями, например, всякие биржи фрилансеров, сайты, на которых можно принять участие в онлайн-играх, некоторые сайты, предназначенные для поддержки общения пользователей между собой, с равной долей отнести и к онлайн-сервисам, например, это относится к сайтам icq или skype. Потенциал развития этого сектора очень велик, и не исключено, что новые интересные возможности для организации контактов и общения через интернет будут придуманы в самое ближайшее время.

**3) сайты электронной коммерции**. Какие типы сайтов выделяются в этой группе? Конечно, самым многочисленным видом будут всякие интернет-магазины, через которые сейчас можно продавать практически всё, что угодно. Также велик список сайтов, предоставляющих различные платные услуги, сейчас с помощью интернета можно оплатить и мобильную связь, и консультацию психолога, и хостинг для сайта. В эту группу войдут и сайты электронных платёжных систем, сайты банков, предоставляющих возможность управления счетами через интернет, сайты обменных пунктов различных валют, сайты, дающие возможность играть на Форексе и т. п. В особую категорию здесь можно вынести сайты, построенные на основе т. н. партнёрских программ, т. е., проще говоря, клоны известных интернет-магазинов, продающие в интернете те же товары той же торговой структуры, и получающие на этом без особых усилий какой-то процент с продаж.

**4) сайты, предоставляющие онлайн-сервисы**. Тут мы обнаружим также большое разнообразие. Некоторые сервисы можно отнести и к другим группам, например, сервисы, предоставляющие хостинг (как хостинг для сайтов, так и хостинг для гостевых, форумов, чатов, блогов, файлов и т. п.) Весьма распространёнными ещё со времён зарождения интернета являются сервисы бесплатной электронной почты. Вообще, многие онлайн-сервисы трудно классифицировать потому, что при всей их огромной популярности и значимости для интернета, главные сайты, предоставляющие их, существуют всего в нескольких экземплярах. Например, одними из важнейших сервисов, необходимость в которых возникла ещё на заре развития интернета, являются поисковые сервисы. Но при этом достаточно популярных среди них (в рунете) всего три. Самым первым поисковиком Рунета был Рамблер. Сегодня лучшим и крупнейшим из поисковых сервисов в Рунете является Яндекс, а в мировом интернете лидерство держит Гугл (Google). Другие интересные сервисы — это, например, сервисы, предоставляющие онлайн-перевод веб-страниц (скажем, translate.ru), сервисы, помогающие вам проверить текст на орфографические ошибки (скажем, orfo.ru), сервисы, позволяющие собирать закладки на интересные вам сайты (скажем, memori.ru или Яндекс.Закладки) и т. д. и т. п. В последнее время, особенно в западном интернете появилась тенденция пытаться создавать онлайн-версии популярных оффлайновых программ, таких, как Word, Excel и т. п. Насколько успешной будет эта попытка переноса в онлайн работы с приложениями, покажет время.

Впрочем, есть и такие виды сайтов, которые изначально не предназначены для живых пользователей. Для чего же они предназначены? Об этом можно узнать, прочитав о **классификации неестественных сайтов.**

Неестественные сайты. Что же это за феномен? Изначально все сайты, появляющиеся в интернете, были предназначены для живых пользователей сети. Какой был смысл тратить кому-то время, усилия и деньги на создание и размещение в сети сайта, который был бы никому не интересен? Однако затем, когда в сети обострилась конкуренция между сайтами и их владельцы стали бороться за вывод своего сайта в «топ», т .е. за то, чтобы пользователи находили их сайт на первых местах в выдаче поисковых систем, и особенно, когда в интернет пришли деньги, появились всякие нехорошие люди, которые стали замусоривать интернет никчёмными и никому (кроме них самих) не нужными сайтами.

Какие цели преследуют создатели этих никчёмных сайтов? Основных целей две. Во-первых, это продвижение другого сайта, либо своего, либо того, владелец которого заплатил за продвижение. Во-вторых, это заработок, получаемый непосредственно с этих самых никчёмных сайтов.

Начнём с сайтов, создаваемых для раскрутки других сайтов:

**1) дорвеи;** Трудно раскрутить один сайт так, чтобы он занимал высокие позиции по множеству поисковых запросов. Поэтому, когда конкуренция в интернете обострилась, недобросовестные вебмастера, чтобы затащить как можно больше посетителей на свой сайт, стали создавать в большом количестве дополнительные сайты, оптимизированные под несколько ключевых слов или фраз, чтобы занять по этим фразам место в топе поисковиков. После того, как пользователь заходил на сайт-дорвей, то он не только не находил там, что искал, а зачастую сразу же автоматически перенаправлялся на главный сайт, тот самый, на который владелец хотел затащить посетителей. Поисковые системы объявили дорвеи, как средство обмана посетителей, вне закона и начали с ними беспощадную борьбу, в результате чего к сегодняшнему моменту пик дорвестроения спал, но они всё ещё иногда встречаются. Существуют даже специальные прграммы, которые позволяют генерировать дорвеи тысячами, заполняя их разнообразным (но оптимизированным под определённые запросы), бредом.

**2) каталоги;** Изначально (когда сайтов в интернете было ещё немного) каталоги создавались с благой целью — дать ссылки посетителям каталога на интересные ресурсы, помочь им быстрее найти нужную информацию в интернете, нужный сайт. Полезные каталоги существуют и сейчас. Однако большая часть каталогов создаётся и используется совсем для других целей. Вебмастерам регистрация сайта в каталогах нужна, чтобы получить больше ссылок на свой сайт и, за счёт этого, приподнять его ближе к топу поисковиков. Владельцы же каталогов, в который они собирают все подряд сайты, зарабатывают на рекламе, либо создают много каталогов, а затем регистрируют в них за деньги чужие сайты. Раньше раскрутка сайта при помощи каталогов давала хороший эффект, но в последнее время этот эффект стал очень слаб. Определённой реинкарнацией каталогов стало появление нового типа каталогов — каталогов статей, которые заполняются уже не просто голыми ссылками, а многочисленными статьями, с проставленными внутри них ссылками на другие сайты.

**3) сателлиты;** Сателлит — английское слово, которое в переводе на русский означает «спутник». Сателлит — это вспомогательный сайт, который не является самодостаточным ресурсом, а предназначен для продвижения главного сайта. Его отличие от дорвея в том, что главной его задачей является не сбор и перенаправление посетителей, а увеличение веса главного сайта при помощи проставленных на него ссылок. Сателлиты могут использоваться как для продвижения собственных сайтов, так и для продвижения чужих сайтов, когда ссылки с них продаются за деньги (в этом случае их можно отнести к категории MFS-сайтов).

Продолжим с сайтов, создаваемых теми, кто решил непосредственно с их помощью заполучить заработок в интернете:

**4) MFA-сайты.** MFA расшифровывается как «Made For Adsense», т. е. «Сделанные Для Адсенсе». Adsense — это популярная система контекстной рекламы Google Adsense. Таким образом, MFA-сайты — это сайты, сделанные для заработка на контекстной рекламе (в рунете, кроме Adsense, имеющей заморочки с выводом средств, распространены и другие системы контекстной рекламы — Бегун и Яндекс.Директ). Конечно, нет ничего плохого, если владелец интересного, оригинального сайта получает дополнительный доход от этого сайта с помощью контекстной рекламы. Однако те, кто делает MFA-сайты, не заморачиваются с созданием нормальных сайтов, содержащих уникальную информацию и действительно нужных пользователям. Часто содержание таких сайтов представляет собой перемешанный копипаст с других сайтов или даже заимствуется с других сайтов и лент новостей в автоматическом режиме. В результате интернет заполняется, по сути, информационным мусором.

**5) MFS-сайты.** MFS расшифровывается как «Made For Sape», а Sape — это наиболее популярная в рунете биржа ссылок. Т. е. MFS-сайт — это сайт, предназначенный для продажи ссылок. В отличие от MFA-сайтов, для MFS-сайтов не нужно привлекать и заводить на сайт посетителей, которые кликали бы на контекстную рекламу, достаточно нарастить сайту «пузомерки», т. е. показатели ТИЦ и PR, чтобы была возможность продавать ссылки с него подороже.

**1.4 Проектирование web-сайта**

Проектирование и разработка сайтов включает: утверждение первоначального технического задания на разработку сайта. Определение структурной схемы сайта расположение разделов, контента и навигации. Веб дизайн создание графических элементов макета сайта, стилей и элементов навигации. Разработка программного кода, модулей, базы данных и других элементов сайта необходимых в проекте. Тестирование и размещение сайта в сети Интернет.

**1.5 Этапы разработки web-сайта**

Разработка web-сайта включает следующие этапы:

1 этап — определение целей создания сайта. Это самый важный этап в создании сайта, потому что нельзя добиться цели, если её нет или если она определена неправильно. От целей будет зависеть весь дальнейший процесс создания вашего сайта, каждый его этап. Поэтому к постановке цели нужно подходить с максимальной серьёзностью.

2 этап — проведение исследований по теме. Прежде, чем преступить к разработке, необходимо проанализировать тему, изучить сайты потенциальных конкурентов. В дальнейшем это поможет в создании собственной концепции.

3 этап — определение типа сайта, разработка технического задания и структуры. На этом этапе разработчик совместно с заказчиком составляет ТЗ (техническое задание) на создание сайта. Техническое задание, как минимум, должно включать следующие пункты:

* тип сайта (портал, промо, визитка, корпоративный, магазин и т.д.);
* функционал сайта (поиск, каталог, лента новостей и т.д.);
* стиль дизайна (строгий, веселый, мрачный и т.д.);
* структура сайта (какие страницы должны быть на сайте);
* структура страниц (какие блоки должны быть на страницах и в каком виде).

При составлении техническом задании обязательно нужно помнить о целях создания сайта. Если какая-нибудь функция сайта не способствует достижению поставленных целей — необходимо от неё отказаться: на сайте она будет лишней.

4 этап — разработка макета дизайна сайта. Этот этап делится на несколько подэтапов:

* генерация идей дизайна;
* разработка предварительного макета дизайна главной страницы;
* исправление замечаний заказчика (если есть), доработка макета до завершенного вида;
* разработка внутренних страниц по аналогичному алгоритму.

В макете должны быть прорисованы все блоки, которые будут на сайте. Если к моменту разработки дизайна текстовые и графические материалы ещё не готовы, можно использовать любую информацию для заполнения, но нельзя оставлять в макете «пустые» места.

Каждый шаг разработки макета дизайна должен выполняться в соответствии с техническими заданиями и целями сайта.

5 этап — html-css вёрстка. В зависимости от целей и задач сайта, верстка должна удовлетворять некоторым требованиям. Обычно эти требования такие:

* кроссбраузерность — страницы должны одинаково отображаться в разных браузерах (Mozilla Firefox, Google Chrome, Opera, Internet Explorer, Safari и т.д.);
* гибкость вёрстки — возможность легко добавлять/удалять информацию на страницы;
* быстрота обработки кода браузером;
* валидность — соответствие стандартам;
* семантическая корректность — логичное и правильное использование элементов HTML.

6 этап — заполнение сайта контентом (информацией). На этом этапе очень важен качественный, профессиональный копирайтинг. Все материалы сайта, будь то тексты или графика, также должны вписываться в общую концепцию сайта, соответствовать его целям и задачам.

7 этап — тестирование сайта и исправление ошибок. Тестирование сайта может проводить как разработчик, так и заказчик. Наилучший вариант — это совместное тестирование.

8 этап — публикация сайта в интернете. На этом этапе сайт размещается на выбранном доменном имени, регистрируется в крупных поисковиках и каталогах.

9 этап — продвижение сайта и реклама в интернете. Когда сайт полностью готов к работе — нужно привлекать на него посетителей. Для «раскрутки» сайта можно воспользоваться контекстной или баннерной рекламой, SEO, SMO и другими методами.

Многие Web-дизайнеры сходятся во мнении, что одна из главных проблем Web-дизайна – многообразие браузеров и платформ, каждая из которых по-разному поддерживает HTML и сценарии. С выпуском каждого нового браузера улучшаются их характеристики и возможности, но это не означает, что более ранние версии при этом исчезают. В большинстве своем люди не склонны гнаться за новейшим и лучшим. Одни довольствуются тем, что у них имеется, а другие, вероятнее всего, работают на компьютерах фирм или учреждений, которые выбрали браузеры за них.

В Web-дизайне нет жестких правил. Поскольку главная задача – сделать содержимое страницы доступным для максимального количества пользователей, то для продвижения вперед одинаково важны и эксперимент, и использование новых технологий с учетом существующих реалий. Залог успеха дизайнерского решения лежит в понимании потребностей аудитории и в четком представлении, как сайт будет использован.

**1.6 Постановка задачи при проектировании web-сайта**

Во всех случаях процессу непосредственной сайта предшествует выработка четкой мотивации, осознание необходимости существования интернет - ресурса. В определенное время человек начинает понимать, что наличие сайта становится необходимостью то ли для реализации какой-нибудь бизнес идеи, то ли для поиска и сплочения вокруг себя единомышленников в сфере определенного интереса, увлечения. Осознав проблему, человек начинает заниматься поиском путей ее реализации. Очень важно, чтоб на этом этапе уже были сформированы конкретные задачи и цели намеченного проекта. В данном случае сайт является не маркетинговым инструментом, а автоматизированной информационной системой. В контексте планируемого бизнес проекта к основным целям относятся:

* создание удобной и надежной интернет-платформы для технического обеспечения учебного процесса;
* ликвидация ошибок из-за человеческого фактора;
* автоматизация рутинной работы;
* поддержка возможности удаленного обучения и общения на электронном портале;
* создание возможности для студентов отслеживать свои результаты в доступном и интересном формате (инфографика, рейтинг и т.д.).

Когда уже намечены конкретные цели, следует приступать к формированию задач, создать план конкретных действий, направленных на достижение поставленных целей. Главное внимание следует обратить на:

* продумывание бизнес-логики, реализации функций на сайте таким образом, чтобы максимально сократить количество выполняемых пользователем операций;
* удобный и интуитивно понятный интерфейс;
* разработку дополнительных привлекательных и полезных пользователю сервисов;
* создание условий для общения (наличие форумов, блогов, досок объявления);
* реализацию маркетингового инструментария (наличие функций обратной связи, организация опросов, анкетирования, тестирования).

**1.7 Выбор программного средства для разработки**

HTML (от англ.  *HyperText Markup Language* — «язык гипертекстовой разметки») — стандартизированный язык разметки документов во Всемирной паутине. Большинство веб-страниц содержат описание разметки на языке HTML (или XHTML). Язык HTML интерпретируется браузерами; полученный в результате интерпретации форматированный текст отображается на экране монитора компьютера или мобильного устройства.

Во всемирной паутине HTML-страницы, как правило, передаются браузерам от сервера по протоколам HTTP или HTTPS, в виде простого текста или с использованием шифрования.

Изначально язык HTML был задуман и создан как средство структурирования и форматирования документов без их привязки к средствам воспроизведения (отображения). В идеале, текст с разметкой HTML должен был без стилистических и структурных искажений воспроизводиться на оборудовании с различной технической оснащённостью (цветной экран современного компьютера, монохромный экран органайзера, ограниченный по размерам экран мобильного телефона или устройства и программы голосового воспроизведения текстов). Однако современное применение HTML очень далеко от его изначальной задачи. Например, тег <table> предназначен для создания в документах таблиц, но иногда используется и для оформления размещения элементов на странице. С течением времени основная идея платформонезависимости языка HTML была принесена в жертву современным потребностям в мультимедийном и графическом оформлении.

CSS (англ. *Cascading Style Sheets* — *каскадные таблицы стилей*) — формальный язык описания внешнего вида документа, написанного с использованием языка разметки.

Преимущественно используется как средство описания, оформления внешнего вида веб-страниц, написанных с помощью языков разметки HTML и XHTML, но может также применяться к любым XML-документам, например, к SVG или XUL.

CSS используется создателями веб-страниц для задания цветов, шрифтов, стилей, расположения отдельных блоков и других аспектов представления внешнего вида этих веб-страниц. Основной целью разработки CSS являлось отделение описания логической структуры веб-страницы (которое производится с помощью HTML или других языков разметки) от описания внешнего вида этой веб-страницы (которое теперь производится с помощью формального языка CSS). Такое разделение может увеличить доступность документа, предоставить большую гибкость и возможность управления его представлением, а также уменьшить сложность и повторяемость в структурном содержимом.

Кроме того, CSS позволяет представлять один и тот же документ в различных стилях или методах вывода, таких как экранное представление, печатное представление, чтение голосом (специальным голосовым браузером или программой чтения с экрана), или при выводе устройствами, использующими шрифт Брайля.

PHP (англ. *PHP: Hypertext Preprocessor* — «PHP: препроцессор гипертекста»; первоначально *Personal Home Page Tools* — «Инструменты для создания персональных веб-страниц») — скриптовый язык общего назначения, интенсивно применяемый для разработки веб-приложений. В настоящее время поддерживается подавляющим большинством хостинг-провайдеров и является одним из лидеров среди языков, применяющихся для создания динамических веб-сайтов.

Язык и его интерпретатор (Zend Engine) разрабатываются группой энтузиастов в рамках проекта с открытым кодом. Проект распространяется под собственной лицензией, несовместимой с GNU GPL.

В области веб-программирования, в частности серверной части, PHP — один из популярных сценарных языков (наряду с JSP, Perl и языками, используемыми в ASP.NET).

Популярность в области построения веб-сайтов определяется наличием большого набора встроенных средств и дополнительных модулей для разработки веб-приложений. Основные из них:

- автоматическое извлечение POST- и GET-параметров, а также переменных окружения веб-сервера в предопределённые массивы;

- взаимодействие с большим количеством различных систем управления базами данных;

- автоматизированная отправка HTTP-заголовков;

- работа с HTTP-авторизацией;

- работа с cookies и сессиями;

- работа с локальными и удалёнными файлами, сокетами;

- обработка файлов, загружаемых на сервер;

- работа с XForms.

В настоящее время PHP используется сотнями тысяч разработчиков. Согласно рейтингу корпорации TIOBE, базирующемуся на данных поисковых систем, в мае 2016 года PHP находился на 6 месте среди языков программирования. К крупнейшим сайтам, использующим PHP, относятся Facebook, Wikipedia и др.

Входит в LAMP — распространённый набор программного обеспечения для создания и хостинга веб-сайтов (Linux, Apache, MySQL, PHP).

Laravel — бесплатный веб-фреймворк с открытым кодом, предназначенный для разработки с использованием архитектурной модели MVC (англ. Model View Controller — модель-представление-контроллер). Laravel выпущен под лицензией MIT.

Ключевые особенности, лежащие в основе архитектуры Laravel:

- пакеты (англ. packages) — позволяют создавать и подключать модули в формате Composer к приложению на Laravel. Многие дополнительные возможности уже доступны в виде таких модулей.

- Eloquent ORM — реализация шаблона проектирования ActiveRecord на PHP. Позволяет строго определить отношения между объектами базы данных. Стандартный для Laravel построитель запросов Fluent поддерживается ядром Eloquent.

- логика приложения — часть разрабатываемого приложения, объявленная либо при помощи контроллеров, либо маршрутов (функций-замыканий). Синтаксис объявлений похож на синтаксис, используемый в каркасе Sinatra.

- обратная маршрутизация связывает между собой генерируемые приложением ссылки и маршруты, позволяя изменять последние с автоматическим обновлением связанных ссылок. При создании ссылок с помощью именованных маршрутов Laravel автоматически генерирует конечные URL.

- REST-контроллеры — дополнительный слой для разделения логики обработки GET- и POST-запросов HTTP.

- автозагрузка классов — механизм автоматической загрузки классов PHP без необходимости подключать файлы их определений в include. Загрузка по требованию предотвращает загрузку ненужных компонентов; загружаются только те из них, которые действительно используются.

- составители представлений (англ. view composers) — блоки кода, которые выполняются при генерации представления (шаблона).

- инверсия управления (англ. Inversion of Control) — позволяет получать экземпляры объектов по принципу обратного управления. Также может использоваться для создания и получения объектов-одиночек (англ. singleton).

- миграции — система управления версиями для баз данных. Позволяет связывать изменения в коде приложения с изменениями, которые требуется внести в структуру БД, что упрощает развёртывание и обновление приложения.

- модульное тестирование (юнит-тесты) — играет очень большую роль в Laravel, который сам по себе содержит большое число тестов для предотвращения регрессий (ошибок вследствие обновления кода или исправления других ошибок).

- страничный вывод (англ. pagination) — упрощает генерацию страниц, заменяя различные способы решения этой задачи единым механизмом, встроенным в Laravel.

- поддержка noSQL СУБД Redis

- множество готовых админ-панелей, шаблонов и CRUD

- шаблонизатор Blade (по умолчанию). Можно также использовать другой аналог - Twig (как в Symfony 4)

- возможность подключать CSS шаблоны.

JavaScript (аббр. JS) — мультипарадигменный язык программирования. Поддерживает объектно-ориентированный, императивный и функциональный стили. Является реализацией стандарта ECMAScript (стандарт ECMA-262).

JavaScript обычно используется как встраиваемый язык для программного доступа к объектам приложений. Наиболее широкое применение находит в браузерах как язык сценариев для придания интерактивности веб-страницам.

Основные архитектурные черты: динамическая типизация, слабая типизация, автоматическое управление памятью, прототипное программирование, функции как объекты первого класса.

Структурно JavaScript можно представить в виде объединения трёх чётко различимых друг от друга частей:

- ядро (ECMAScript),

- объектная модель браузера (Browser Object Model или BOM)

- объектная модель документа (Document Object Model или DOM).

**1.8 Разработка интерфейса**

Интерфейс пользователя, создаваемый Вами, должен использовать стандартные, привычные пользователям элементы, и обеспечивать максимальное удобство. Всё это, в конечном счёте, определяется таким критерием как эффективность интерфейса - максимальный результат с минимальными усилиями.

Принципы создания удобного интерфейса сайта известны. В качестве самых общих принципов при создании пользовательских интерфейсов можно рассматривать три основных положения:

* сайт должен помогать выполнить задачу, а не становиться этой задачей;
* сайт должен быть максимально понятным и функциональным.

Первый принцип — это так называемая «прозрачность» интерфейса. Интерфейс пользователя должен быть интуитивно понятным, простым для освоения, и не создавать для пользователя проблем, которые он вынужден будет преодолевать в процессе работы. Используйте стандартные, без излишнего украшательства.

Второй принцип заключается в пренебрежении интеллектуальными способностями пользователей. Интерфейс пользователя должен быть максимально дружественным.

Для соблюдения второго принципа не нужно позволять сайту «исправлять» действия пользователя и указывать, что как именно ему действовать, загоняя в узкие рамки. Также не следует чрезмерно увлекаться выводом информационных сообщений-подсказок, особенно диалоговых, это отвлекает пользователя от работы.

Для разработки интерфейса использовался движок шаблонов Blade, CSS-фреймворк Bootstrap 4, JavaScript, в том числе Vue JS. Вид домашней страницы администратора представлен на рисунке 1.

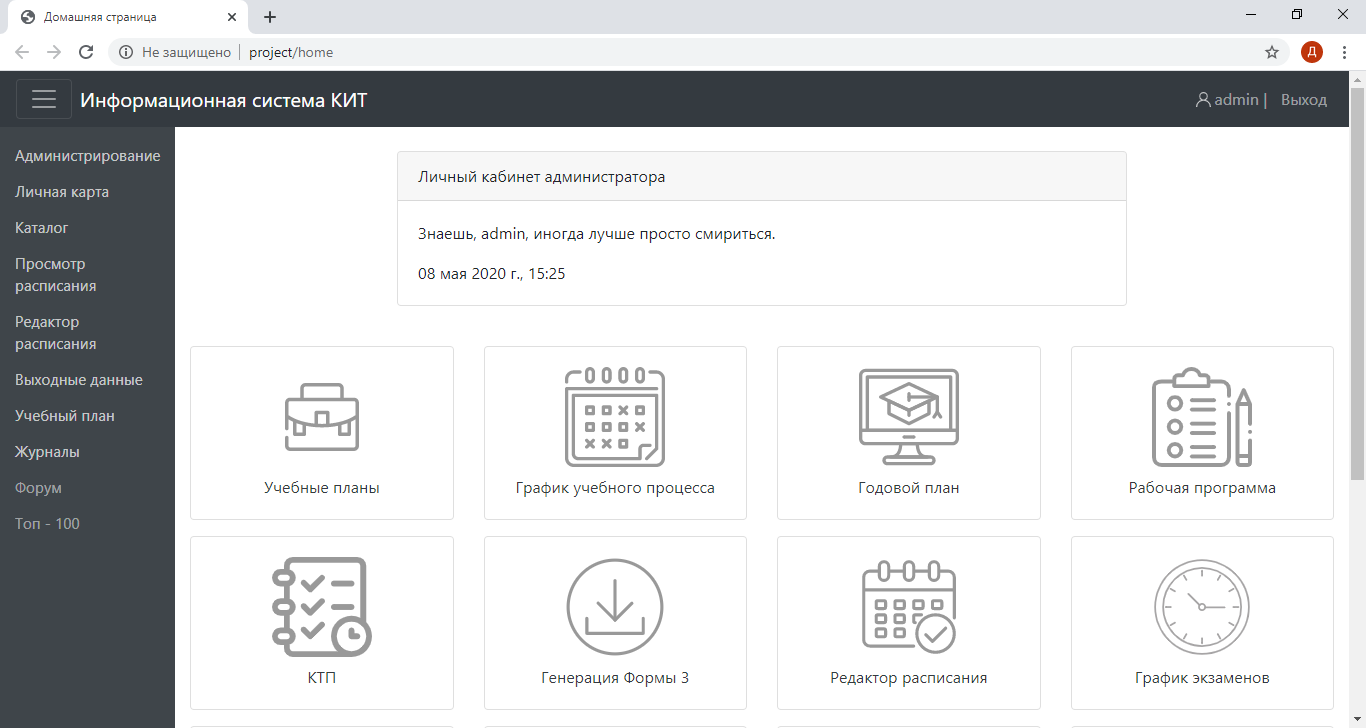


Рисунок 1 – Главная страница сайта.

**1.9 Разработка базы данных**

Тщательное проектирование базы данных чрезвычайно важно для безупречной работы приложения. Как установка принтера в дальнем конце офиса ведет к снижению производительности труда, размещение данных со слабыми взаимосвязями снижает эффективность и может вынудить сервер базы данных тратить значительное время на поиск требуемых данных.

MySQL – это реляционная база данных. Важная особенность реляционных систем, их отличие от одноуровневых баз данных – возможность располагать данные в нескольких таблицах. Взаимосвязанные данные можно хранить в отдельных таблицах и объединять по ключу, общему для обеих таблиц. Ключ – это отношение (relation) между таблицами. Выбор первичного ключа (primary key) – наиболее важное решение, принимаемое при разработке новой базы данных. Самое главное, что следует понимать, – вы должны гарантировать уникальность выбранного ключа. Если есть вероятность того, что значение некоторого атрибута может совпасть у двух записей, то его нельзя использовать в качестве первичного ключа. Если таблица содержит ключевые поля из другой таблицы, то между ними образуется связь – взаимоотношением внешнего ключа (foreign key), например «начальник-подчиненный» или «покупатель-покупка».

Представление о взаимоотношениях данных и наиболее эффективном способе их организации называется нормализацией. Нормализация заключается в разделении данных на основе логических взаимоотношений с целью минимизировать дублирование данных. Повторяющиеся данные понапрасну расходуют дисковое пространство сервера и затрудняют их обслуживание. При внесении изменений в повторяющиеся данные есть риск пропустить какие-то из них, что может привести к возникновению несогласованностей в базе данных.

С другой стороны, лучшее – враг хорошего: когда данные хранятся по частям в отдельных таблицах, это может потребовать слишком больших накладных расходов на их извлечение, да и запросы могут получаться чересчур замысловатыми. Главная цель – найти золотую середину.

Взаимоотношения или связи, в базах данных подразделяются на следующие категории:

* связи "один-к-одному";
* связи "один-ко-многим";
* связи "многие-ко-многим".

**1.10 Разработка административной части сайта**

В более или менее сложной системе пользователи имеют различные роли и разный уровень доступа. В Laravel одним из способов реализации контроля доступа является фасад Gate, с помощью которого можно определить правила, имеющие определнное имя и параметры. Таким образом в любом месте приложения можно вызвать проверку на доступ, а также группировать маршруты по этим правилам.

Самый высокий уровень доступа у группы пользователей Администраторы. Администратор имеет полный контроль над системой, может видеть список других пользователей, создавать новые учетные записи, управлять системой.

При разработке системы, в которой предусмотрено разграничение доступа, особенное внимание стоит уделить безопасности. В данном приложении проверка выполняется при web и api-маршрутизации, в действиях контроллера, шаблонах представлений (например, для условного отображения элементов интерфейса). При попытке доступа к ресурсу, на который у пользователя нет прав, возвращается статус 403 – Forbidden.

**1.11 Обоснование выбора среды разработки**

Laravel - это бесплатный  PHP фреймворк с открытым исходным кодом, созданный Тейлором Отвеллом для разработки веб-приложений по архитектурному шаблону MVC.

Можно сказать, что на появление Laravel повлияли остальные PHP фреймворки.

Он был создан как альтернатива фреймворку Codeigniter, в котором было недостаточно полезных функций для разработки веб-приложений. В качестве основы Laravel выступают компоненты другого фреймворка - Symfony.

С помощью менеджера пакетов Composer, фреймворк Laravel позволяет легко устанавливать и подключать различные компоненты для использования в веб-приложении.

Реализация шаблона ActiveRecord - Eloquent ORM, позволяет установить отношения между объектами базы данных веб-приложения и выстраивать удобные запросы для манипуляции данными.

Механизм автозагрузки классов позволяет не подключать вручную файлы через include и предотвращает загрузку не используемых компонентов.

Удобная система миграций помогает упростить развертывание и обновление веб-приложения.

В Laravel есть встроенная поддержка движка шаблонов Blade, с помощью которого  можно делать простые представления веб-приложения используя специальный синтаксис.

При создании приложения можно использовать Artisan - интерфейс командной строки для ввода встроенных команд, а также создания своих собственных.

Laravel поддерживает такие удобные функции, как: настройка выполнения задач по расписанию, готовая пагинация, механизмы авторизации, втом числе с помощью api-токенов, практически готовые к использованию, разграничение прав доступа. Также имеется большое количество фасадов и хелперов, позволяющих писать лаконичный и понятный код, а также получить данные или вызвать функцию в любом месте приложения и пространстве имен.

В Laravel можно очень быстро написать REST API для интеграции с мобильными приложениями, использования на других сайтах и так далее. Также из коробки доступны Bootstrap 4, VueJS, Webpack. Таким образом, Laravel отлично подходит в качестве удобного, быстрого и надежного средства разработки.

**2 ПРОЕКТНАЯ ЧАСТЬ**

**2.1 Общая структура сайта**

Понятное построение структуры сайта позволяет потенциальному пользователю находить информацию оперативно и легко. Архитектура сайта должна быть проста и интуитивно удобна. Пункты меню сгруппированы по тематическим разделам.

Данный дипломный проект состоит из девяти основных разделов: «Администрирование», «Личная карта», «Каталог», «Просмотр расписания», «Редактор расписания», «Выходные данные», «Журналы», «Топ-100», «Форум» (Рисунок 2).

Каждая страница соответствует определенному маршруту. Доступ к маршрутам определяется по роли пользователя после авторизации в системе.

Сайт

Администрирование

Личная карта

Список пользователей

Добавить пользователя

Форум

Расписание

Каталог

Студенты

Специальности

Основное расписание

Преподаватели

Группы

Изменения в расписании

Аудиторный фонд

Журналы

Учебный план

График экзаменов

Дисциплины

Журналы успеваемости

Рабочий учебный план

Праздничные дни

Циклы дисциплин

Выходные данные

Текущая успеваемость

График учебного процесса

Форма 2

Годовой план

Аттестация

Рабочая учебная программа

Форма 3

Календарно-тематический план

Нагрузка преподавателя

Рисунок 2 - Структура сайта.

**2.2 Разработка интерфейса web-сайта**

При открытии сайта появляется начальная страница сайта с меню с пунктами, соответствующими роли пользователя. Пункты меню сгруппированы по разделам, также в навигации показано текущее положение на сайте. На домашней странице собрана информация и ссылки для быстрого доступа. У студента это прогресс закрытия сессии, у преподавателя процент выставления оценок за текущий семестр. На Рисунке 3 показана домашняя страница студента. В меню также имеются дополнительные пункты, соответствующие роли, например, расписание на сегодня, зачетка, список студентов группы (если есть кураторство), личная динамика.

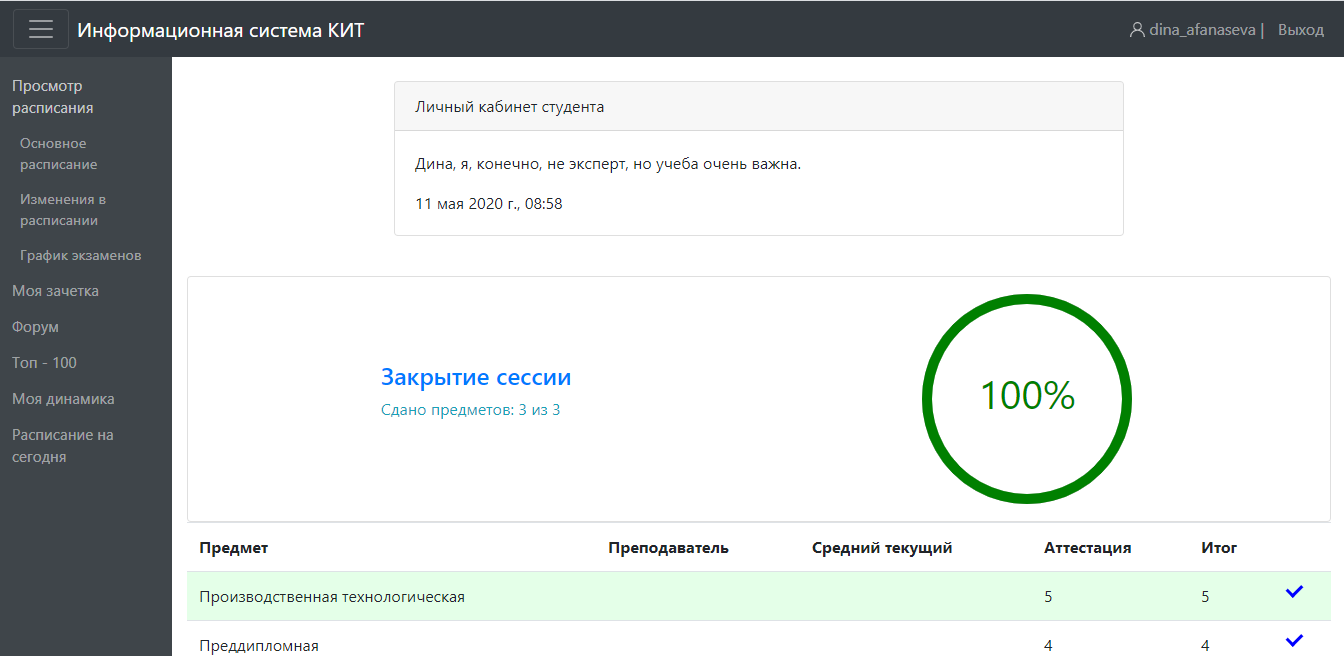


Рисунок 3 – Домашняя страница студента.

**2.3 Дизайн web-сайта**

В данном проекте создан минималистичный деловой интерфейс с использованием CSS-фреймворка Bootstrap 4. Страницы стилизованы таким образом, чтобы обеспечить удобочитаемость, понятность и комфортную работу пользователя.

Цветная окраска веб-сайта сдержанная и лаконичная, цвета элементов интерфейса логически обоснованы и помогают донести информацию до пользователя. Стиль кнопок и уведомлений соответствует сути выполняемого действия и един для всего сайта.

Позиция элементов в иерархии документа и их значение выделяется цветом, размером, отступом группировкой.

В разделе Администрирование имеются 2 вкладки – «Список пользователей, где отображены данные пользователей системы и их роли и «Добавить пользователя», где можно ввести в систему нового администратора, заведующего или диспетчера (Рисунок 4).

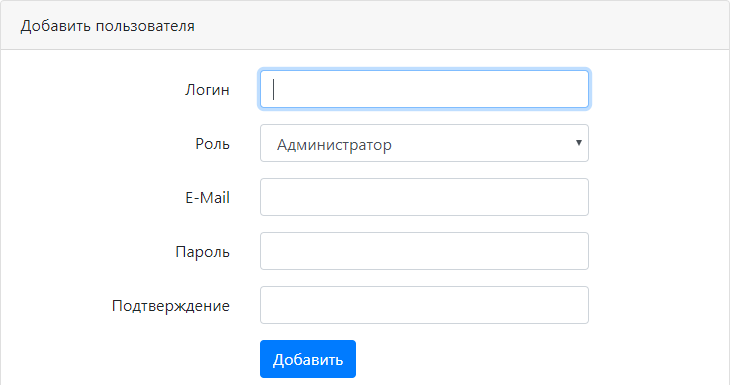


Рисунок 4 – Регистрация пользователя.

В разделе Личная карта есть 2 пункта – «Студенты» и «Преподаватели». На странице «Студенты» (Рисунок 5) отображен список студентов колледжа, также имеется фильтр по специальности, курсу, базе, группе, языку обучения, форме оплаты и поиск по ФИО. Напротив каждой записи есть ссылки на личную карту и зачетку студента.

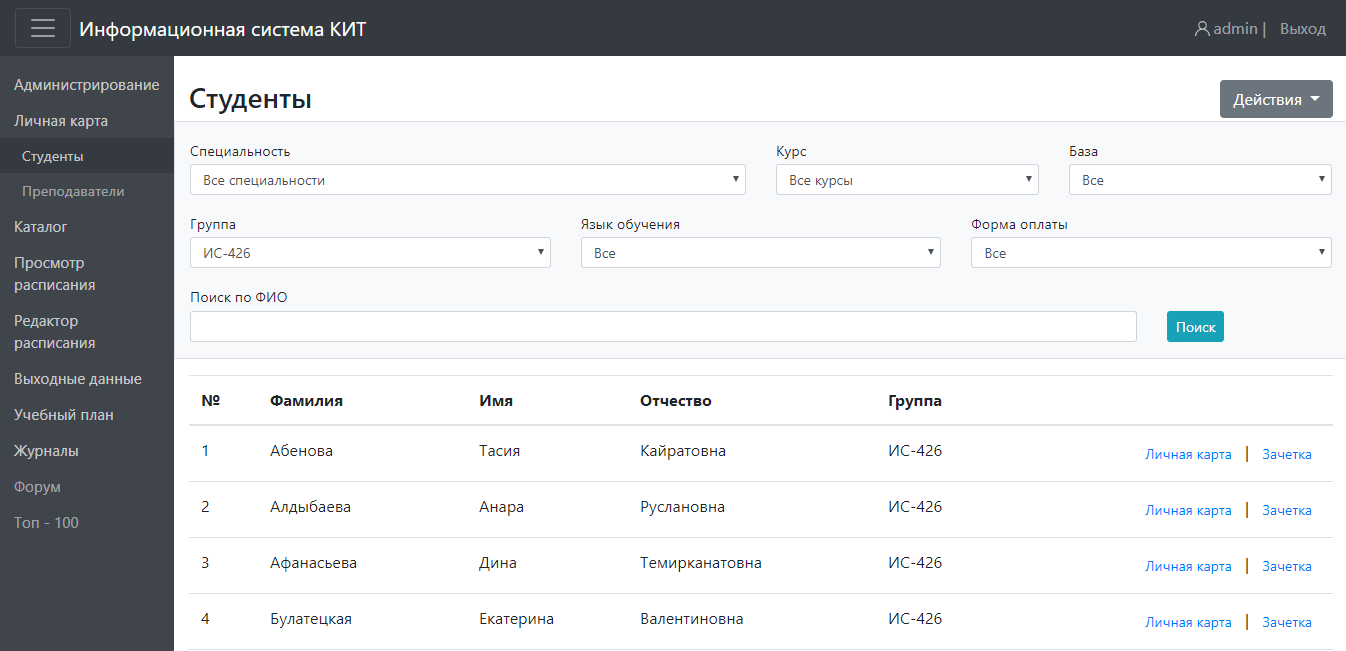


Рисунок 5 – Список студентов.

При нажатии на кнопку «Действия» доступные следующие опции:

- «Добавить» - откроется страница добавления нового студента;

- «Загрузить» - появится модальное окно с формой выбора файла .xlsx для загрузки. Файл должен иметь следующие столбцы: Фамилия, Имя, Отчество, Дата рождения;

- «Создать учетки» - перед каждой строкой появится чекбокс. Необходимо отметить нужных студентов или нажать «Выбрать все» и нажать «Создать учетки». Для выбранных студентов будут сгенерированы учетные записи и на экране появятся данные для входа (Рисунок 6).

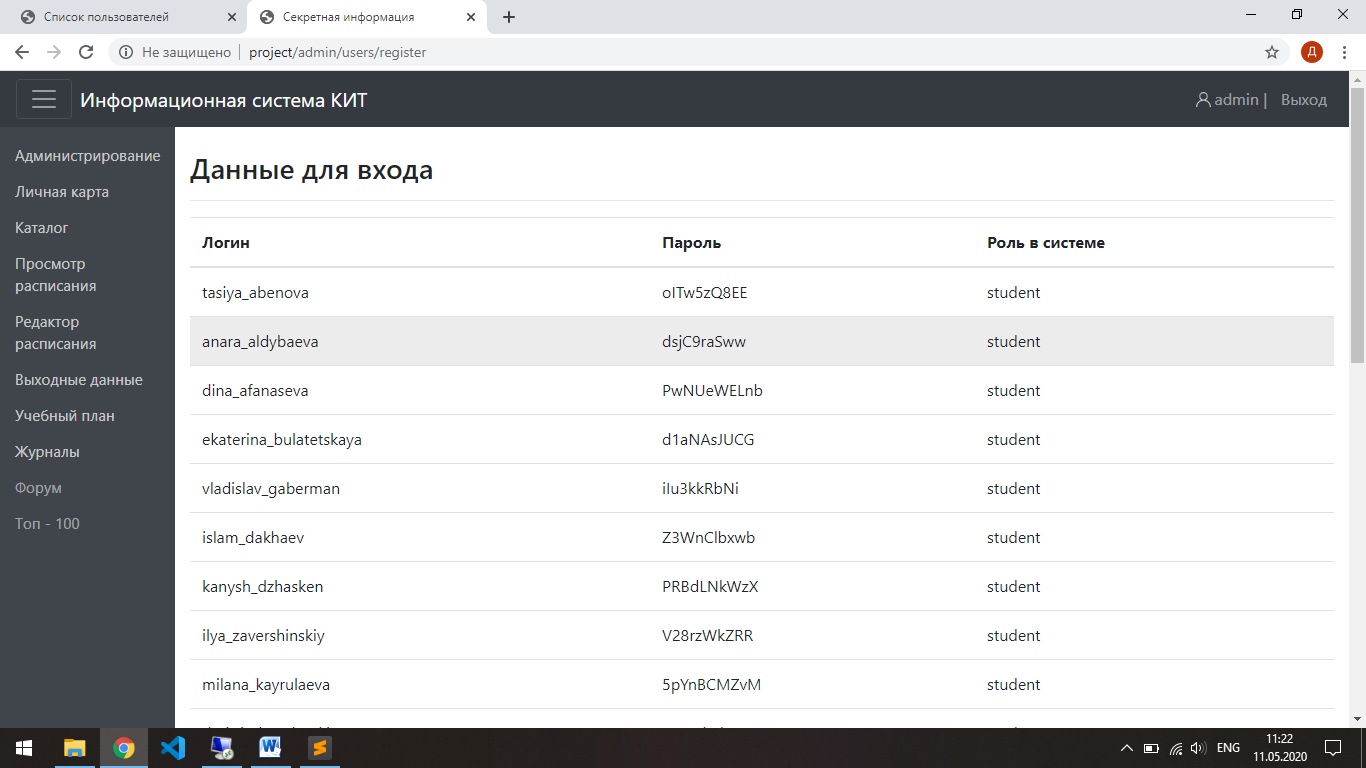


Рисунок 6 – Генерация учетных данных.

При переходе по ссылке «Личная карта» напротив записи откроется личная карта студента (Рисунок 7), в которой можно редактировать личные сведения и сведения об образовании, а также загрузить фото.

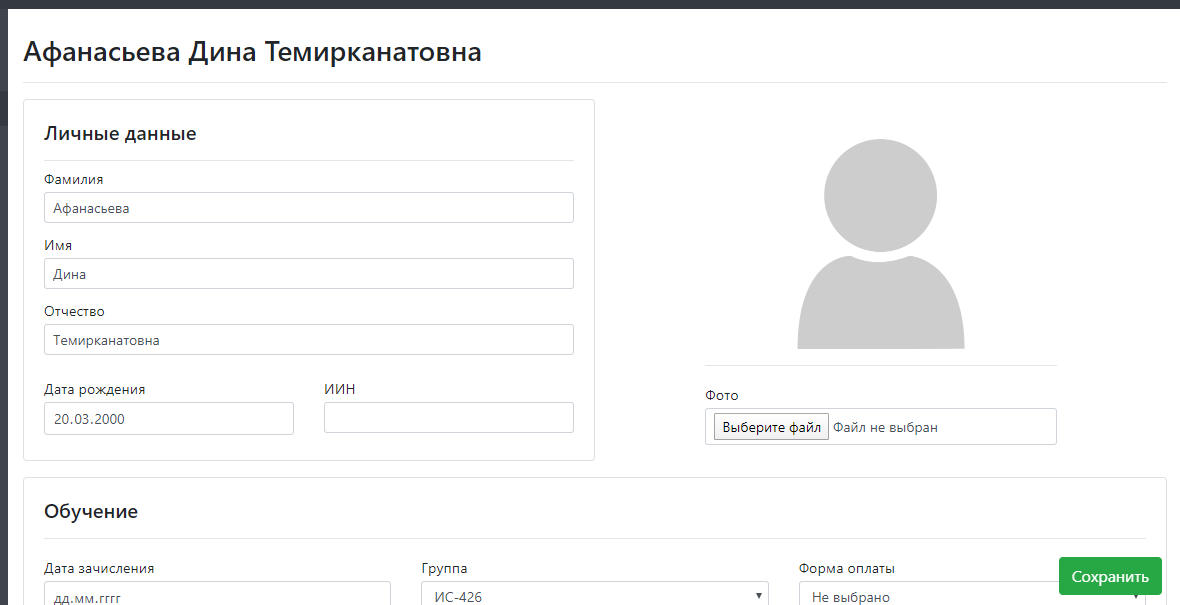


Рисунок 7 – Личная карта студента.

При переходе по ссылке «Зачетка» откроется зачетка студента с вкладками, соответствующими прошедшим семестрам (Рисунок 8).

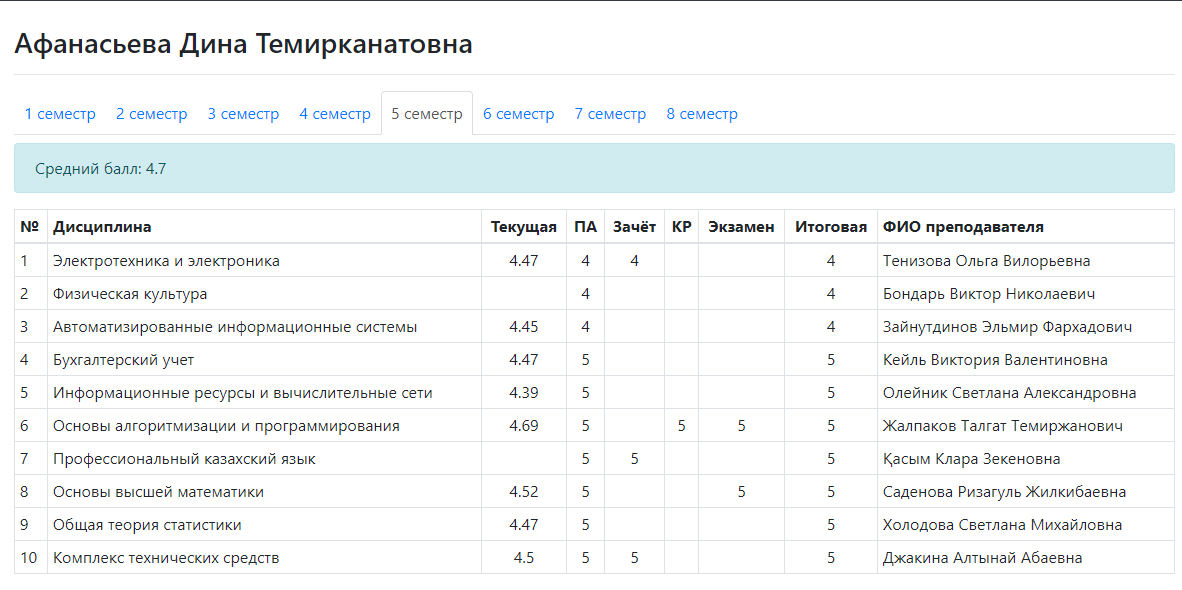


Рисунок 8 – Зачетная книжка.

Функционал модуля «Преподаватели» аналогичен вышеописанному.

В разделе меню «Каталог» содержатся следующие справочники: «Специальности», «Группы», «Аудиторный фонд», «Дисциплины», «Циклы дисциплин» (Рисунки 9 – 13).

На странице списка групп имеется фильтр по специальности, курсу, базе и языку обучения. Для группы указывается название, база, год поступления и выпуска, курс, специальность, язык обучения и куратор. Ссылка «Студенты» открывает список студентов данной группы, где можно также добавить нового, создать учетные записи и сформировать подгруппы.

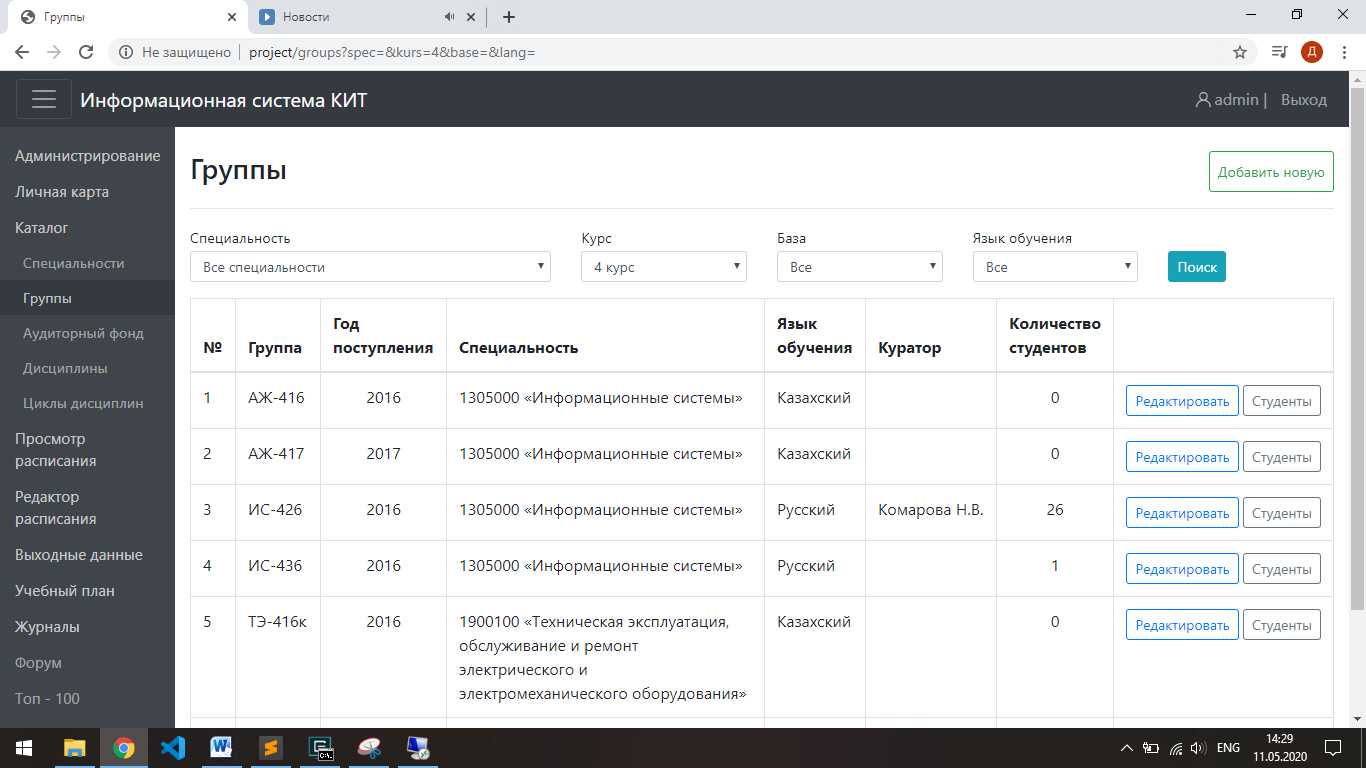


Рисунок 9 – Список групп.

Чтобы добавить или отредактировать специальность, нужно ввести шифр, названия на русском и казахском языках и указать отделение.

|  |  |
| --- | --- |
| Рисунок 10 – Специальности. | Рисунок 11 – Кабинеты. |

В справочнике аудиторий указывается номер, название, корпус, вместимость и описание.

При редактировании дисциплины указываются название и сокращение на 2 языках и признак делимости: не формировать подгруппы, формировать всегда или только на практики. Это нужно для корректного формирования учебного плана и расписания. При нажатии кнопки «Загрузить» открывается модальное окно выбора файла для загрузки списка дисциплин.

Для циклов дисциплин указываются название и сокращение на 2 языках.

|  |  |
| --- | --- |
| Рисунок 12 – Дисциплины. | Рисунок 13 – Циклы дисциплин. |

В разделе «Просмотр расписания» есть ссылки на основное расписание, изменения в расписании на дату и график экзаменов.

На странице «Основное расписание» можно просмотреть сетку расписания, указав учебный год и семестр, также есть фильтр по отделению, курсу и языку обучения (Рисунок 14). Можно просмотреть отдельно расписание группы, преподавателя или занятость аудитории. При нажатии на кнопку «Экспорт в Excel» будет загружен файл с сеткой расписания.

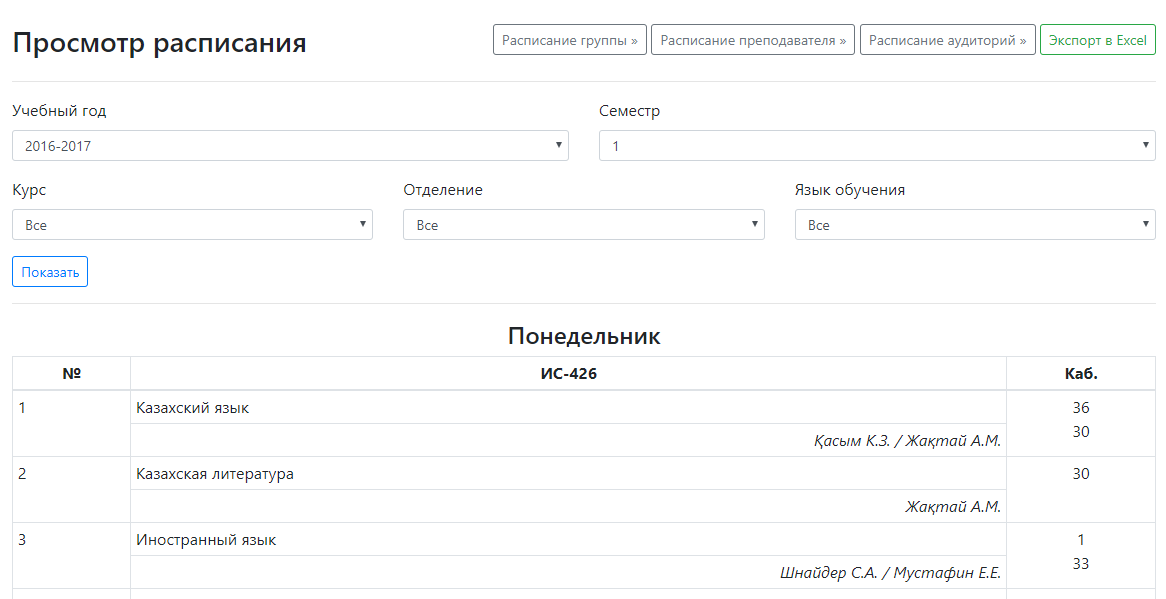


Рисунок 14 – Основное расписание.

На странице «Изменения в расписании» можно просмотреть фактическое расписание на дату с учетом изменений (Рисунок 15). Фильтр аналогичен основному. Также есть возможность посмотреть изменения для группы или преподавателя и распечатать или сохранить в PDF.

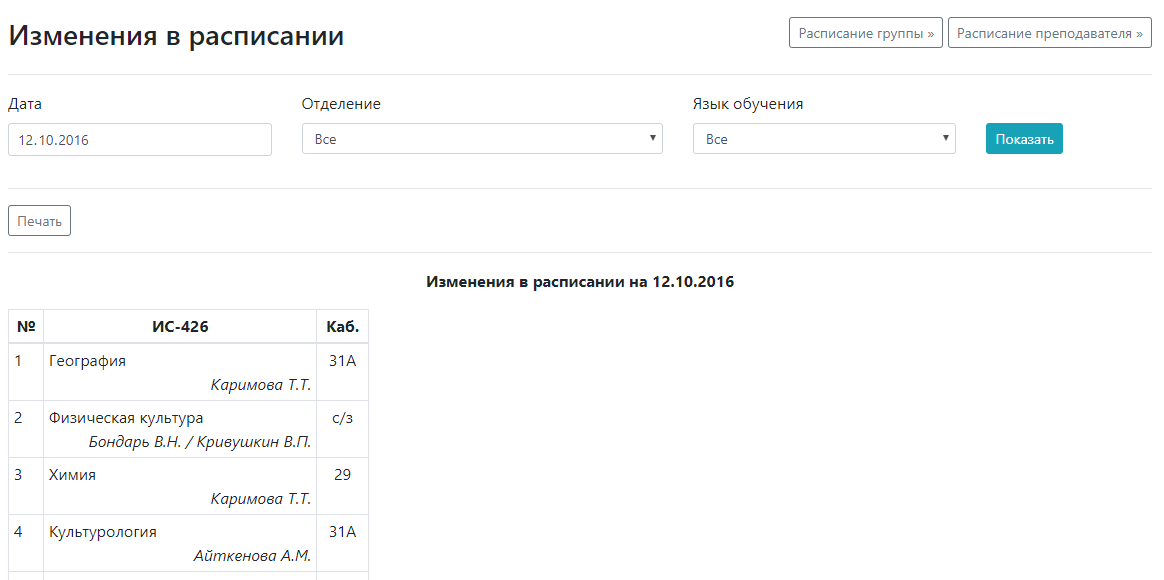


Рисунок 15 – Изменения в расписании.

На странице «График экзаменов» можно просмотреть расписание сессии, выбрав группу и семестр, а также распечатать или сохранить (Рисунок 16).

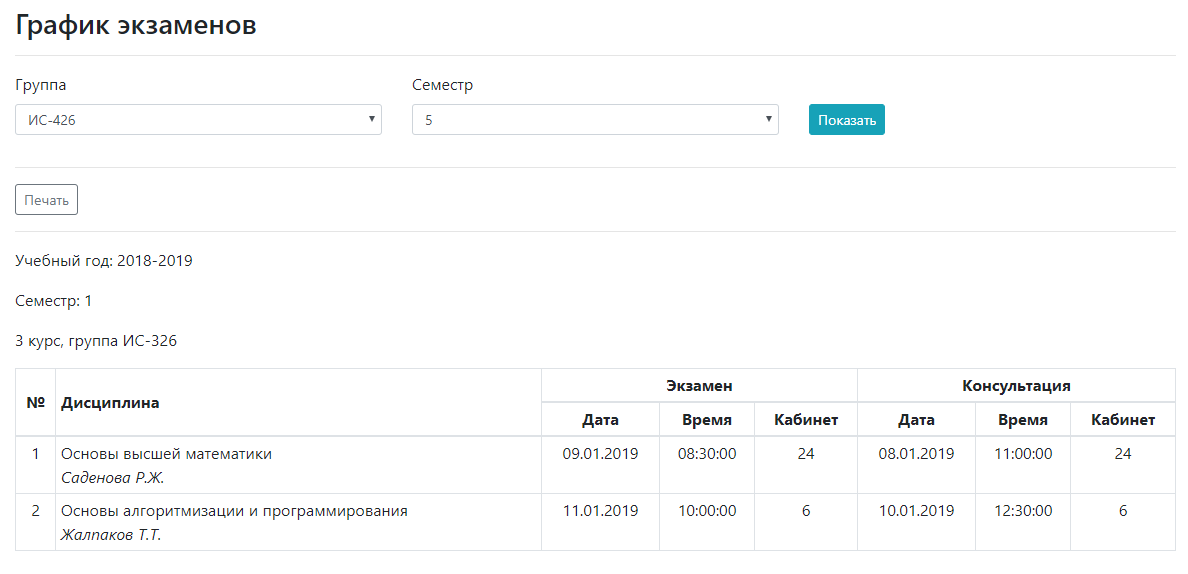


Рисунок 16 – График экзаменов.

В разделе меню «Редактор расписания» собран функционал для диспетчера. На странице «Основное расписание» (Рисунок 17) можно создавать расписание группы на выбранный учебный год и семестр.

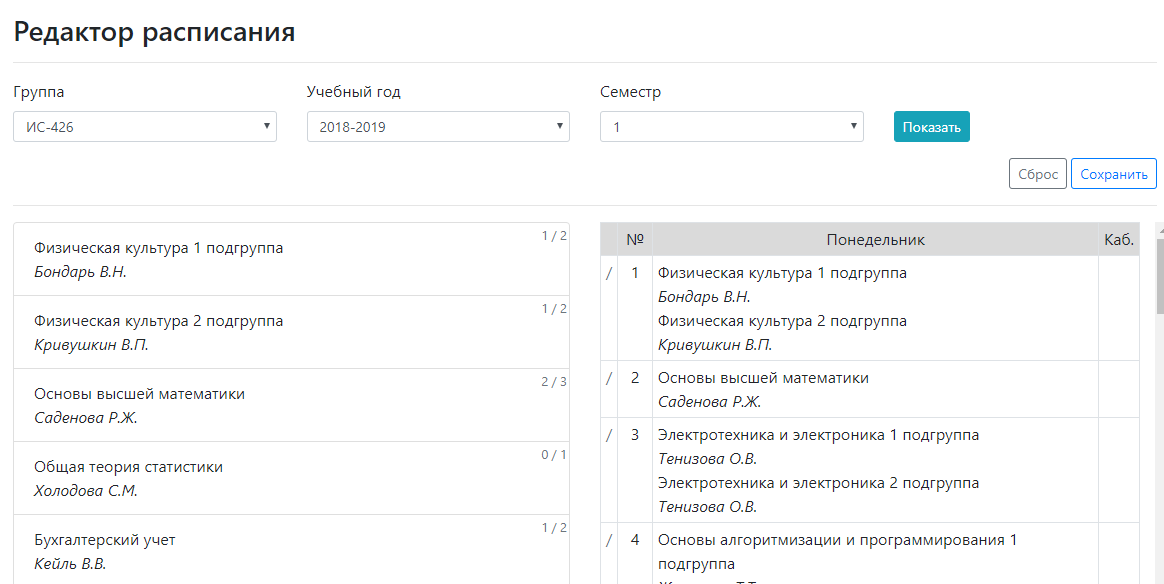


Рисунок 17 – Редактор основного расписания.

В левой части окна отображается список изучаемых дисциплин, в правой – сетка расписания. Дисциплины перетаскиваются мышкой в нужную ячейку таблицы, при делении на числитель/знаменатель ячейку можно разбить. Для назначения кабинета необходимо кликнуть по его ячейке и выбрать кабинет в модальном окне. При формировании расписания показывается количество часов в неделю по плану и количество назначенных в расписание, таким образом, по мере распределения по дням дисциплины пропадают из списка. Если при назначении в расписание возникает наложение, выдается соответствующее сообщение. Также проверяется занятость кабинетов.

На странице «Изменения в расписании» можно внести изменения на определенную дату (Рисунок 18).

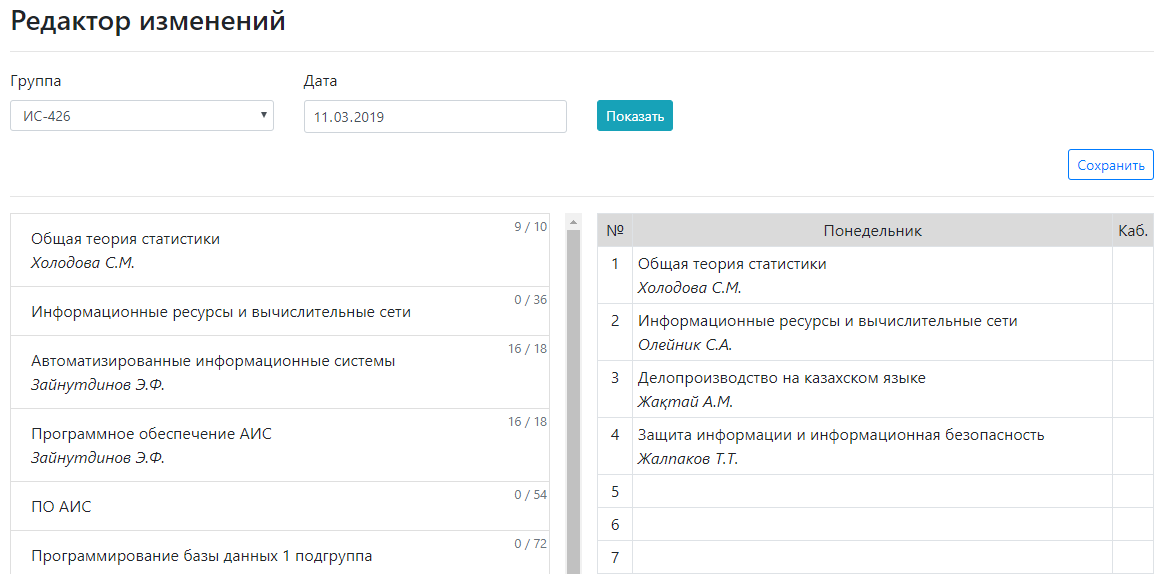


Рисунок 18 – Редактор изменений в расписании.

Весь функционал аналогичен основному, также можно заменить преподавателя, кликнув по имени и выбрав нужного в модальном окне.

На странице «График экзаменов» назначаются даты, время и кабинет для экзаменов и консультаций (Рисунок 19). Необходимо выбрать группу и семестр и отобразятся дисциплины, по которым будет экзамен.

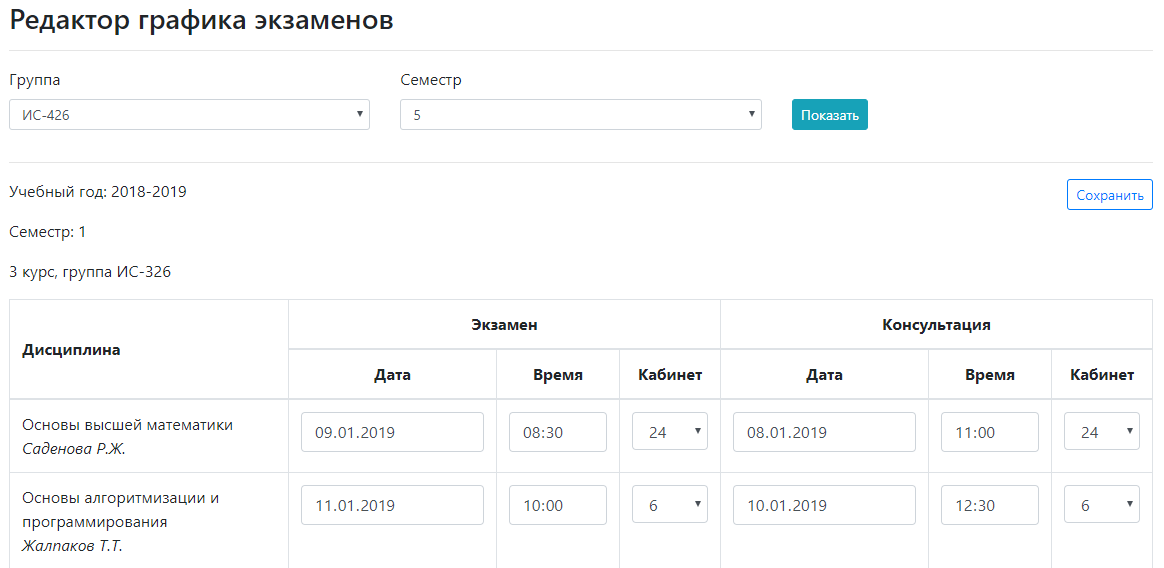


Рисунок 19 – Редактор графика экзаменов.

Также для корректной работы расписания есть модуль «Праздничные дни», где можно добавить даты с названиями. В эти даты уроки не будут назначены.

В разделе меню «Выходные данные» находятся ссылки на выходные документы. Для загрузки Формы 2 нужно выбрать группу, учебный год, семестр и месяц. Для Формы 3 выбирается преподаватель и учебный год. Для просмотра и скачивания нагрузки преподавателя нужно также выбрать ФИО и учебный год (Рисунок 20).

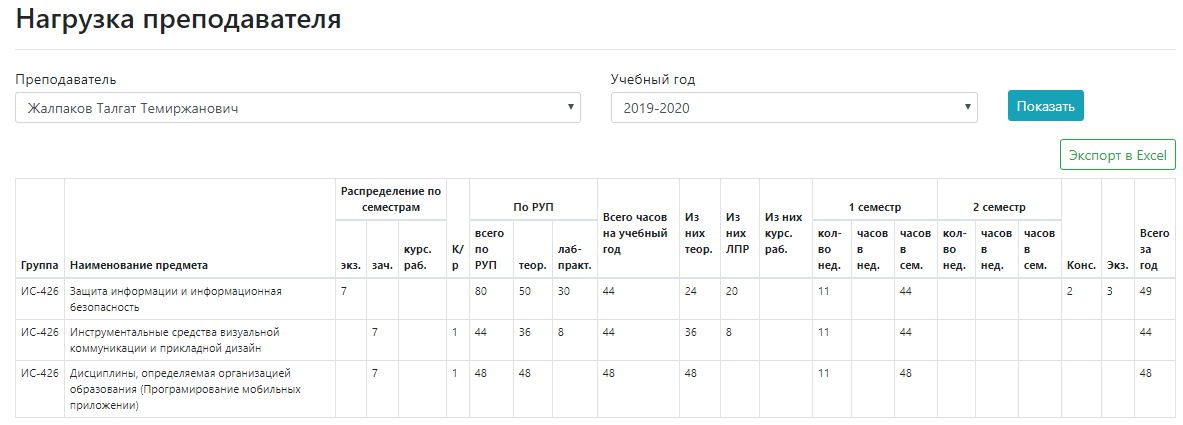


Рисунок 20 – Нагрузка преподавателя.

В разделе меню «Учебный план» имеются следующие модули: «Рабочий учебный план», «График учебного процесса», «Годовой план», «Рабочая учебная программа» и «Календарно-тематический план».

На странице «Рабочий учебный план» формируется учебный план группы на весь период обучения (Рисунок 21).

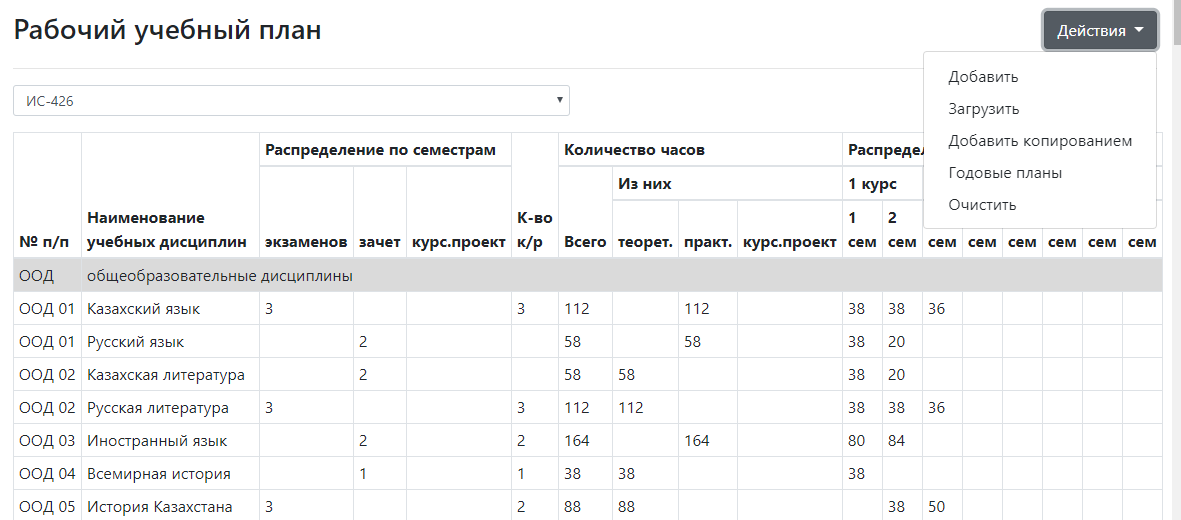


Рисунок 21 – Рабочий учебный план.

Для импорта плана из файла необходимо нажать «Загрузить» и выбрать файл для загрузки. Для копирования из существующего плана нужно нажать «Добавить копированием» и выбрать нужный план. Для ручного добавления дисциплины нажать «Добавить» и заполнить форму.

Для пунктов плана доступно редактирование формы контроля, количества недель практики, распределение часов по видам занятий. Для редактирования нужно нажать на строку таблицы, изменить значения и сохранить (Рисунок 22). Сумма часов автоматически подсчитывается.

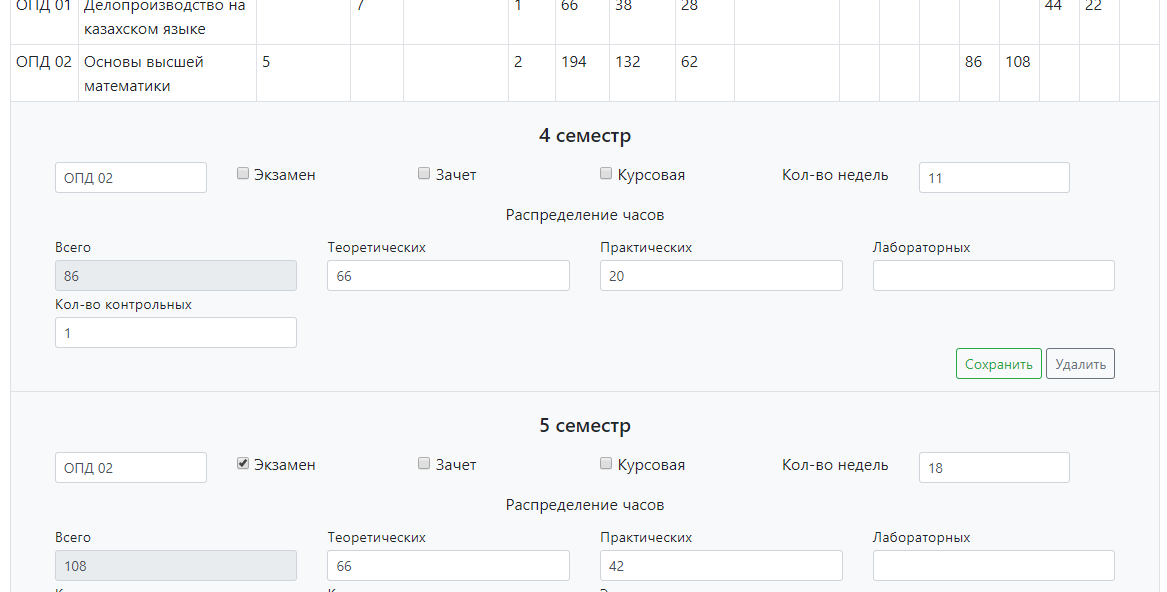


Рисунок 22 – Редактирование плана.

На странице «График учебного процесса» определяются временные рамки семестров и количество недель (общее и теоретическое обучение) (Рисунок 23). На каждый учебный год добавляются графики и к ним прикрепляются группы. У группы может быть только один график на каждый год.

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

Рисунок 23 – График учебного процесса.

На странице «Годовой план» при выборе группы и курса показываются дисциплины данного учебного года по плану (Рисунок 24). В данном модуле выполняется назначение преподавателей на занятия. При нажатии кнопки «Обновить подгруппы» для дисциплин, имеющих признак делимости, появится вторая подгруппа в случае, если количество студентов превышает 24. Количество часов будет распределено в зависимости от того, делится ли по этому предмету группа на все занятия или только на практические. Данный план можно экспортировать в Excel.

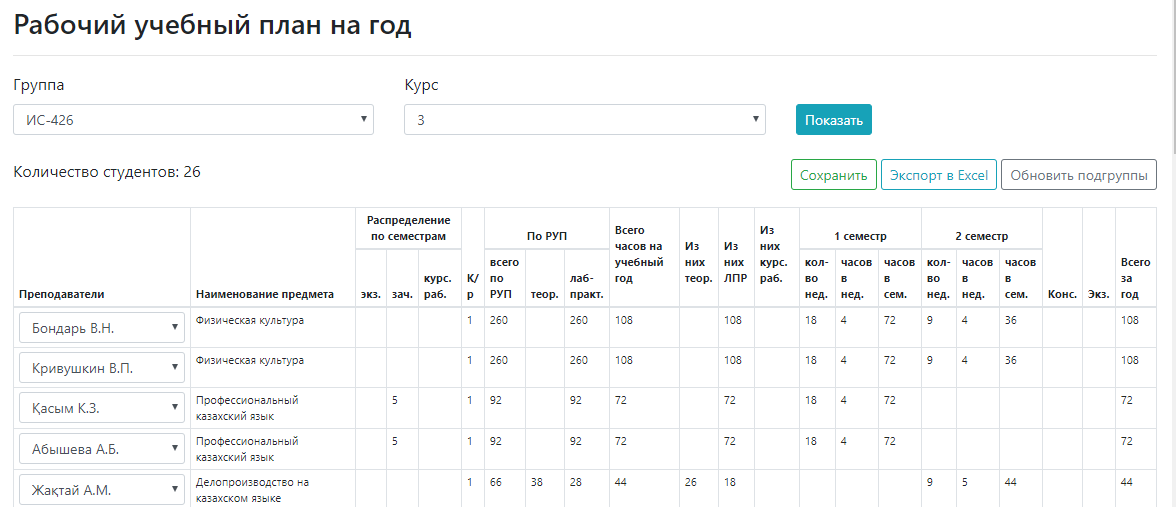


Рисунок 24 – Рабочий учебный план на год.

В разделе «Рабочая учебная программа» формируется тематический план по предмету и определяется количество часов на каждый урок. На начальной странице отображен список дисциплин каждой группы и фильтр по группам и дисциплинам. При выборе дисциплины группы из списка открывается страница с распределением видов занятий по семестрам (Рисунок 25) и редактором тематического плана (Рисунок 26).

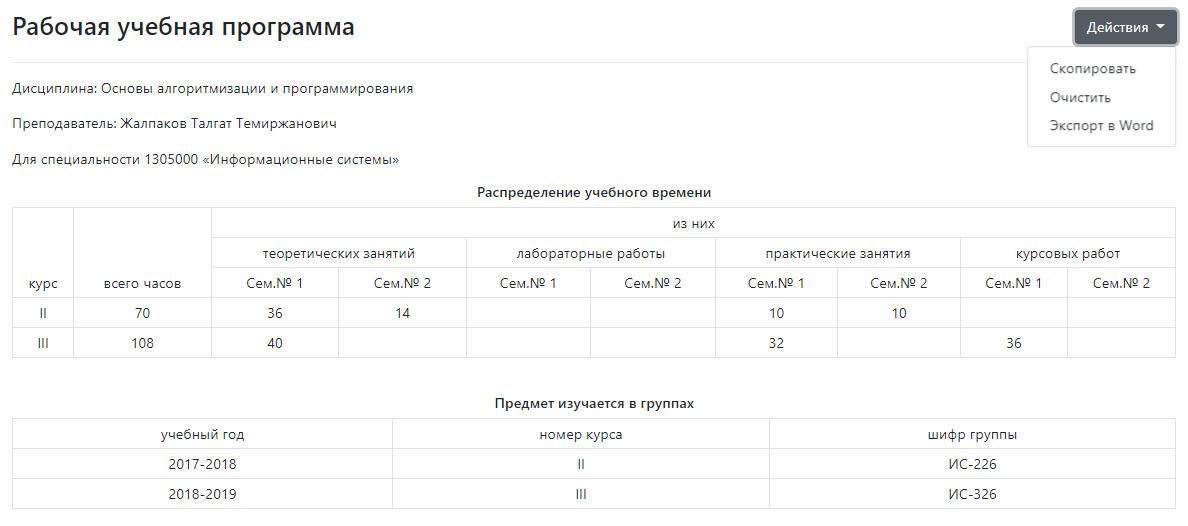


Рисунок 26 – РП (распределение часов)



Рисунок 26 – Редактор тематического плана.

С помощью соответствующих кнопок добавляются и удаляются разделы плана и темы. Для каждой темы определяется общее количество часов (по умолчанию 2) и количество часов ЛПЗ. При этом подсчитывается итог. Если сумма часов в таблице превышает количество часов по плану, добавление новых строк блокируется. Также проверяется количество часов ЛПЗ в каждом семестре и работает ограничение при вводе чисел в таблицу.

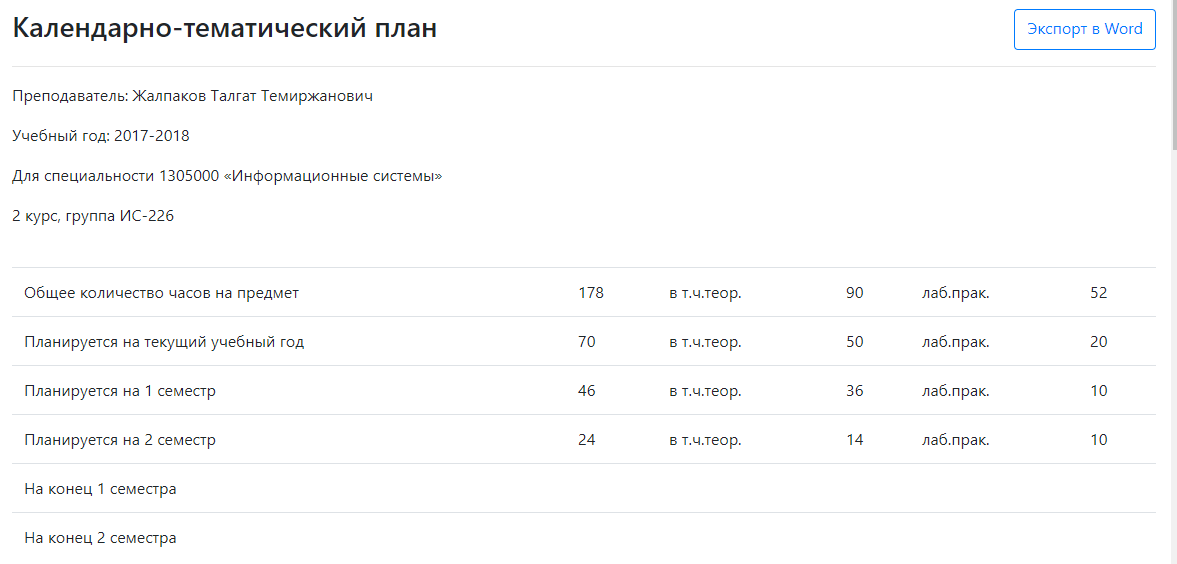
Готовую программу можно скачать в формате .docx.

При нажатии «Скопировать» откроется список групп, также изучающих эту дисциплину по плану. Можно просмотреть каждую из них. Для копирования программы нужно кликнуть «Скопировать» и все темы, разделы и часы перенесутся в целевую группу. Скопированную программу можно впоследствии редактировать.

В разделе «Календарно-тематические планы» можно просматривать и скачивать в формат .docx тематические планы на год, автоматически сформированные из РП (Рисунок 27). Для перехода в план нужно найти его в списке, воспользовавшись фильтром по группе, курсу или дисциплине.

Если группа делится по данной дисциплине, то будут сформированы 2 плана для каждой подгруппы, при этом, если деление только на лабораторно-практические занятия, в КТП второй подгруппы попадут только часы ЛПЗ.

Даты занятий берутся из фактического расписания, тип занятия определяется по наличию часов ЛПЗ.



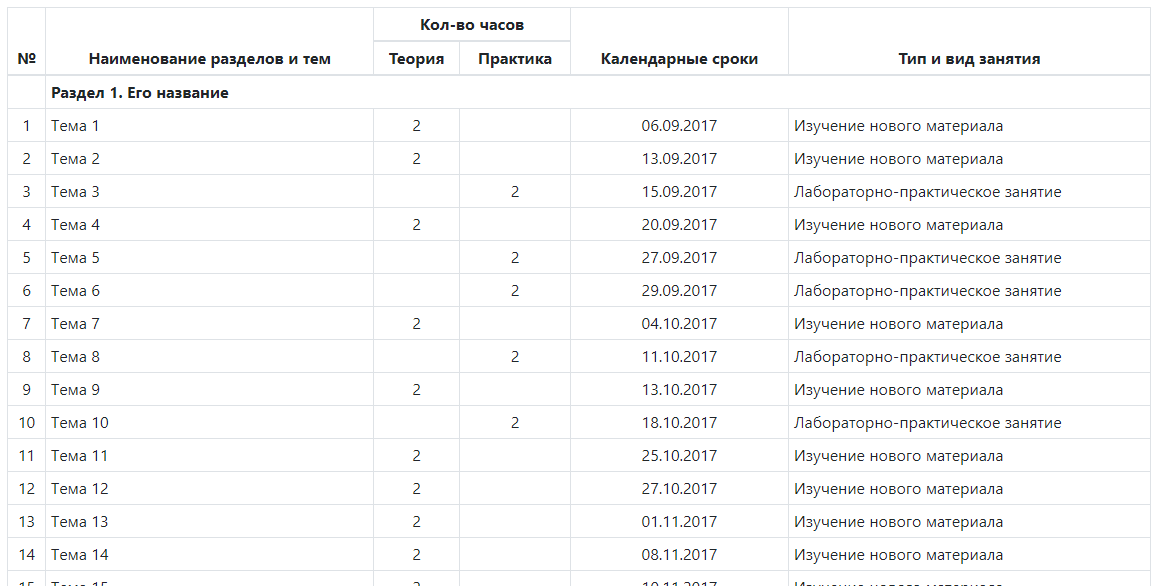


Рисунок 27 – Календарно-тематический план.

В разделе меню «Журналы» имеются следующие пункты: «Журналы успеваемости», «Текущая успеваемость», «Итоговая аттестация».

На странице «Журналы успеваемости» отображается список журналов (Рисунок 28). Если пользователь авторизован как преподаватель, то он видит только свои дисциплины, как и в модуле с тематическими планами.

При нажатии на «Перейти» отображается журнал по данной дисциплине для всей группы или соответствующей подгруппы (Рисунок 29). Если группа делится на практики, в журнале 1 подгруппы на лабораторных занятиях ячейки 2 подгруппы неактивны. Для редактирования доступны ячейки только тех уроков, что были назначены в расписание, т.е. по факту. Оценки вводятся в ячейки таблицы, после чего необходимо нажать «Сохранить». Если студент отстутвовал, вводится «н».

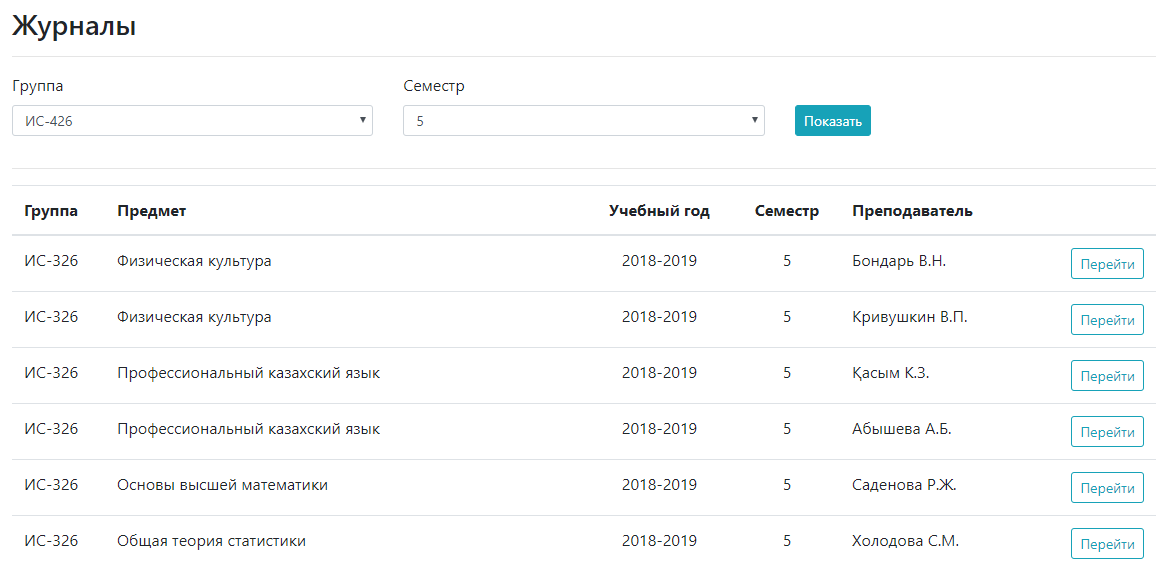


Рисунок 28 – Список журналов.



Рисунок 29 – Журнал.

На странице «Текущая успеваемость» при выборе группы и семестра показываются средние текущие оценки каждого студента по каждому предмету (Рисунок 30).

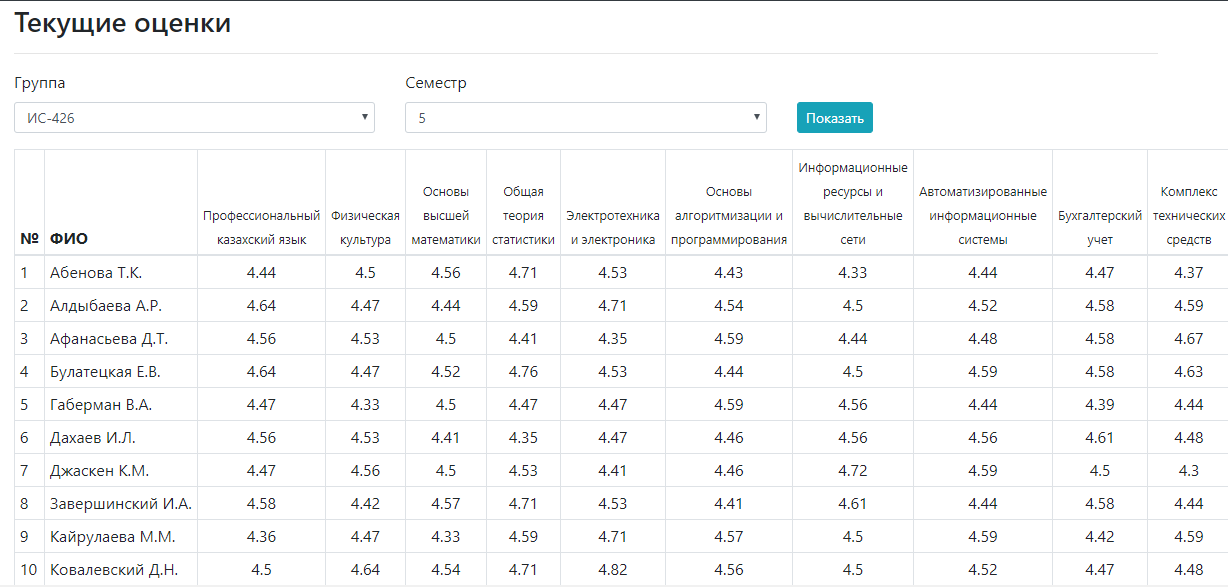


Рисунок 30 – Средний балл.

На странице «Аттестация» при выборе группы и семестра отображен список дисциплин по плану. При переходе к дисциплине открывается список студентов, для каждого отображен средний балл по журналу и выставляется промежуточная аттестация, зачет, экзамен, курсовая (при наличии) и итоговая (Рисунок 31). Эти оценки студент будет видеть в своей зачетке.

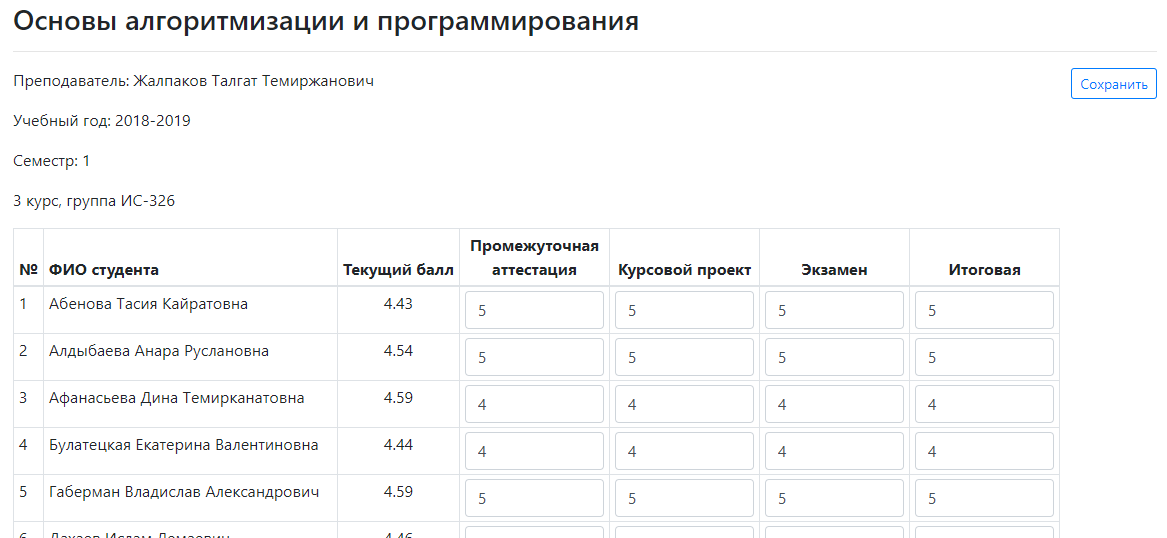


Рисунок 31 – Выставление аттестации.

В разделе «Форум» можно создавать темы (групповые чаты) и общаться. Отображен список тем, к которым авторизованный пользователь имеет доступ (Рисунок 32). Для создания новой темы нужно кликнуть «Создать тему» и ввести название и описание.

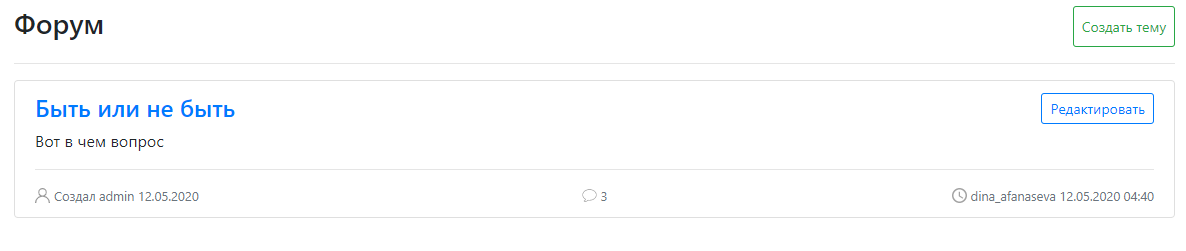


Рисунок 32 – Список тем.

Автор темы может ограничить права доступа. Для этого нужно кликнуть «Редактировать». На открывшейся странице назначаются права доступа для студентов и преподавателей. Для каждой категории пользователей можно установить опции «Все», «Никто» или выбрать галочками определенных пользователей (Рисунок 33). Администраторы по умолчанию имеют доступ ко всем темам.

При переходе в тему открывается окно чата (Рисунок 34). Есть функция ответа на сообщения, а также отправки файлов. При отправке файла определяется его тип. Если это изображение, оно будет отображено в окне чата, в противном случае текст сообщения будет содержать ссылку на скачивание файла.

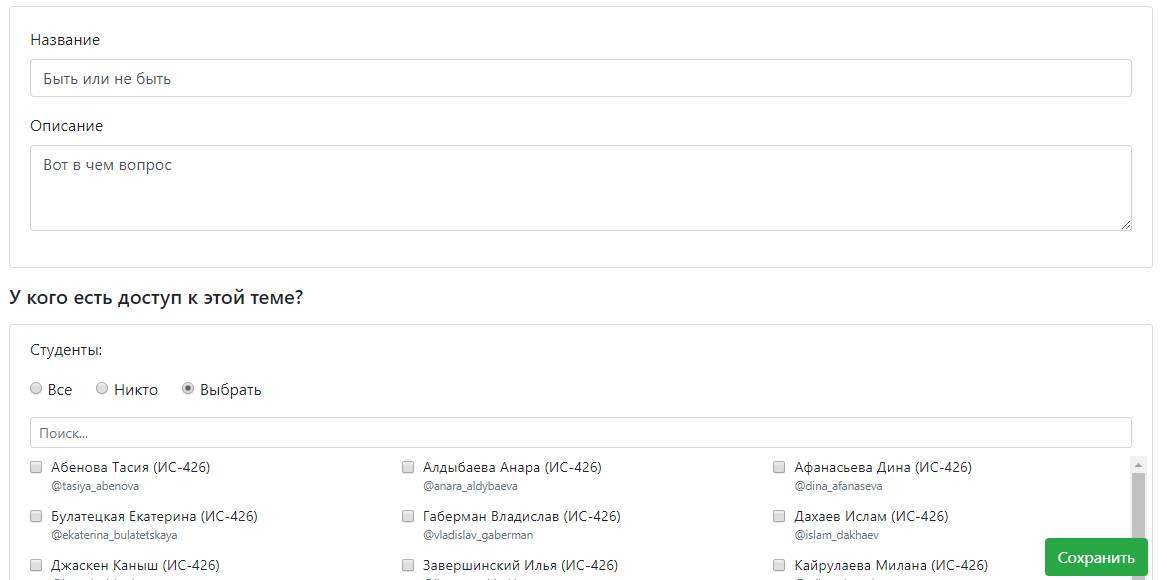


Рисунок 33 – Назначение прав на тему.

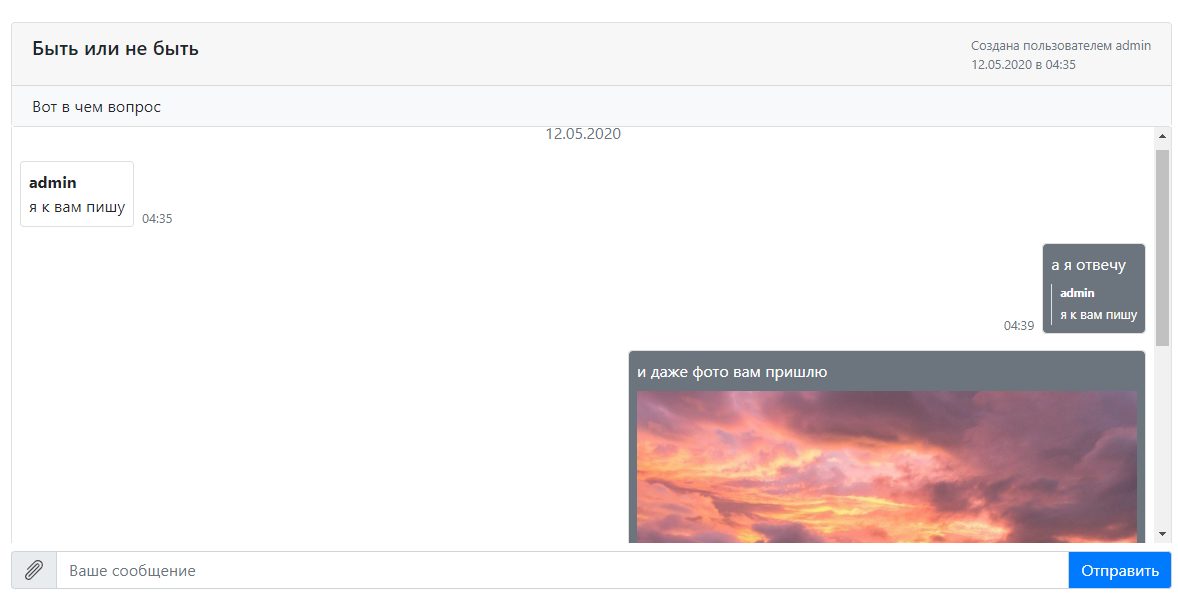


Рисунок 34 – Чат.

На странице «Топ – 100» отображается рейтинг студентов по среднему баллу по итогам семестра (Рисунок 35). Также имеется фильтр по группе.

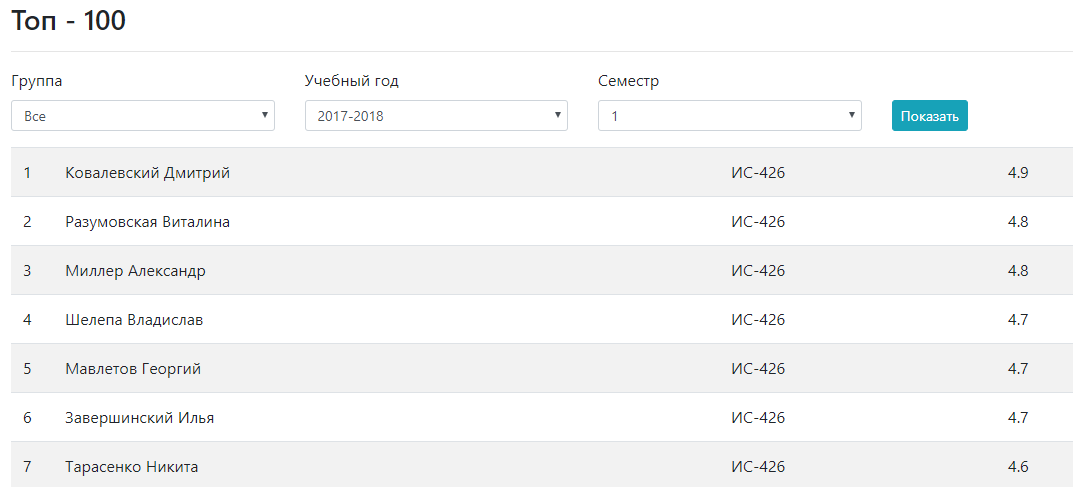


Рисунок 35 – Топ – 100 студентов.

Для пользователя, авторизованного как студент, доступен раздел «Моя динамика». В нем можно просмотреть инфографику по средним баллам и итогу в разрезе семестров (Рисунок 36).

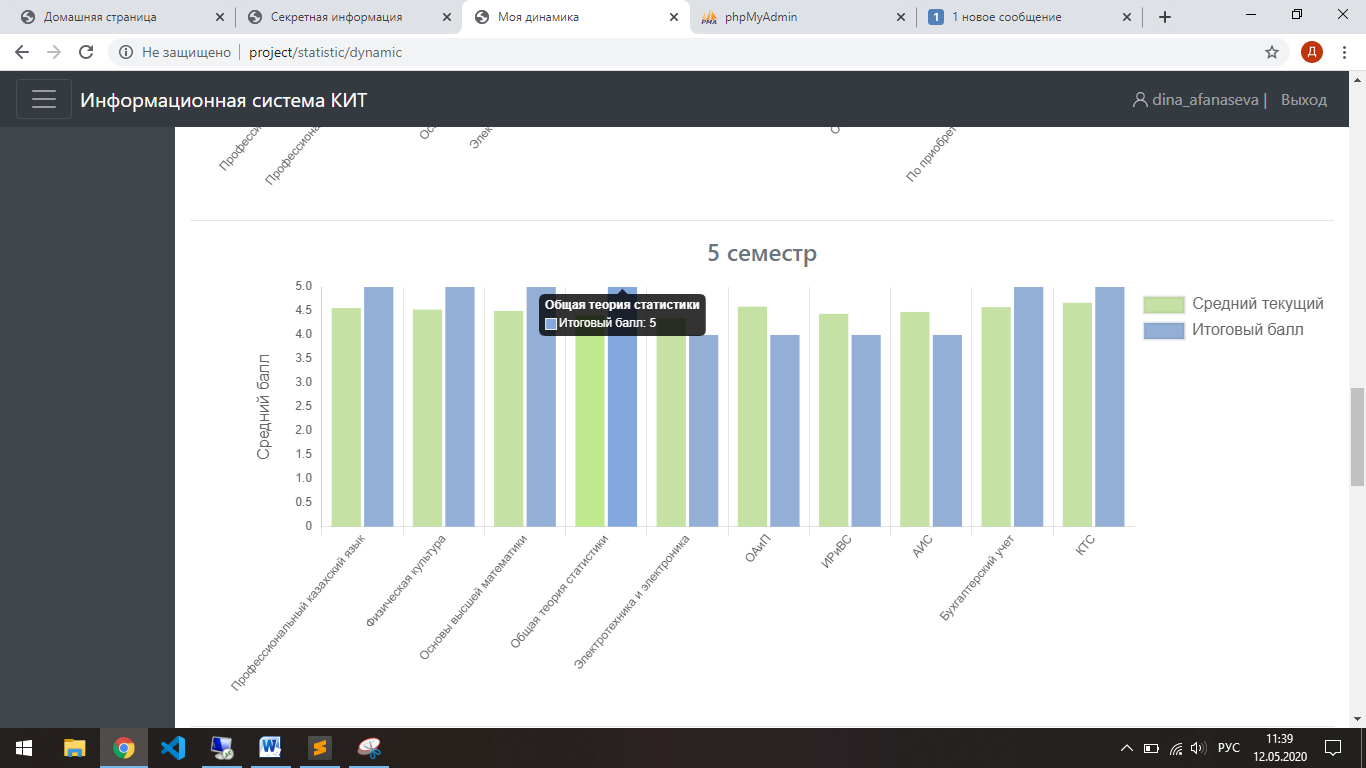


Рисунок 36 – Динамика.

**2.4 Описание создания страниц web-сайта**

Проект в Laravel построен по схеме MVC (Модель – Представление – Контроллер). Модели представляют собой классы, описывающие сущности.

Система объектно-реляционного отображения (ORM) Eloquent — красивая и простая реализация шаблона ActiveRecord в Laravel для работы с базами данных. Каждая таблица имеет соответствующий класс-модель, который используется для работы с этой таблицей. Модели позволяют запрашивать данные из таблиц, а также вставлять в них новые записи.

В коде модели можно задавать дополнительные параметры, вычисляемые свойства, определить, какие поля будут скрыты при сериализации, определить отношения между моделями, создать методы класса и т.д. Пример кода файла Student.php приведен в листинге 1.

Листинг 1 – код модели.

*<?php*

*namespace App;*

*use App\DateConvert;*

*use App\Plan;*

*use App\Result;*

*use Illuminate\Database\Eloquent\Model;*

*class Student extends Model*

*{*

*public $timestamps = false;*

*protected $fillable = ['surname', 'name', 'patronymic', 'group\_id', 'subgroup', 'iin', 'born', 'enter', 'pay\_id', 'address', 'phone', 'email', 'photo'];*

*public function getShortNameAttribute()*

*{*

*return $this->surname.' '.*

*mb\_substr($this->name, 0, 1).'.'.*

*($this->patronymic ? mb\_substr($this->patronymic, 0, 1).'.' : '');*

*}*

*public function getFullNameAttribute()*

*{*

*return $this->surname.' '.*

*($this->name ? ' '.$this->name : '').*

*($this->patronymic ? ' '.$this->patronymic : '');*

*}*

*public function group()*

*{*

*return $this->belongsTo('App\Group')->withDefault();*

*}*

*public function ratings()*

*{*

*return $this->hasMany('App\Rating');*

*}*

*public function avgRating($subject, $semestr)*

*{*

*$avg = $this->ratings()->whereHas('lesson', function($q1) use($subject, $semestr) {*

*$q1->whereHas('plan', function($q2) use($subject, $semestr) {*

*$q2->where('subject\_id', $subject)->where('semestr', $semestr);*

*});*

*})->avg('value');*

*return $avg ? round($avg, 2) : '';*

*}*

*public function results()*

*{*

*return $this->hasMany('App\Result');*

*}*

*public function subjects($semestr)*

*{*

*$subjects = Plan::select('subject\_id', 'semestr')*

*->where('group\_id', $this->group\_id)*

*->where('semestr', $semestr)*

*->whereNotIn('cikl\_id', [7,8,9])*

*->distinct()->get();*

*return $subjects;*

*}*

*}*

Контроллеры содержат основную логику приложения. В них производятся действия с моделями, обработка данных. Контроллер – это класс, каждый метод которого, как правило, обрабатывает HTTP – запрос. Контроллеры могут быть сгруппированы по пространствам имен. Стандартная практика – помещение контроллеров для REST API в папку Api с соответствующим пространством имен. В данном проекте создан API для отображения расписания и отправки сообщений в чат. Это позволит интегрировать проект с другими приложениями и сервисами.

Соответствие между маршрутами и методами контроллеров определяются в файлах web.php и api.php папки routes (листинг 2).

Листинг 2 – маршрутизация приложения.

*…*

*Route::group(['middleware' => ['auth', 'can:admin']], function () {*

*Route::get('/admin/users', 'Admin\UserController@index')->name('admin/users');*

*Route::get('/admin/users/create', 'Admin\UserController@create')->name('admin/users/create');*

*Route::post('/admin/users/register', 'Admin\UserController@register');*

*Route::post('/admin/users', 'Admin\UserController@store');*

*});*

*…*

Метод контроллера возвращает представление (веб-страницу), файл на скачиваение, переадресацию или статус ошибки, например 404 при отсутствии ресурса. Пример метода контроллера в листинге 3.

Листинг 3 – Фрагмент файла StudentController.php.

*…*

*public function edit($id)*

*{*

*$student = Student::findOrFail($id);*

*return view('student.edit', [*

*'student' => $student,*

*'groups' => Group::orderBy('name', 'asc')->get(),*

*'pays' => Pay::orderBy('name', 'asc')->get(),*

*]);*

*}*

*…*

Представления – страницы, возвращаемые при обработке маршрута. Представляют собой файлы шаблона с расширением .blade.php. Данные передаются при рендеринге представления. Синтаксис движка blade имеет большое количество конструкций для отображения страниц, например, @foreach, @if ... @else, @unless, @auth, @guest, @can, @cannot, @forelse. Также можно определить одну или несколько мастер-страниц, общих для нескольких шаблонов. Как правило, они содержат секции подключения стилей, скриптов, меню, подвал и т.д. Подключаются в дочернее представление с помощью конструкции @extends('layouts.app'). В мастер-странице определяются места, куда будет встроен код из файлов. Таких секций может быть несколько, например, title, content, scripts. Пример файла journal/index.blade.php в листинге 4.

Листинг 4 – код представления.

*@extends('layouts.app')*

*@section('title', 'Журналы')*

*@section('content')*

*<h3>Журналы</h3>*

*<hr>*

*<form>*

*<div class="row">*

*<div class="form-group col-sm-4">*

*<label>Группа</label>*

*<select name="group" class="form-control form-control-sm">*

*<option value="">Группа</option>*

*@foreach($groups as $g)*

*<option value="{{ $g->id }}" {{ $g->id == @$\_GET['group'] ? 'selected' : '' }}>{{ $g->name }}</option>*

*@endforeach*

*</select>*

*</div>*

*<div class="form-group col-sm-4">*

*<label>Семестр</label>*

*<select name="sem" class="form-control form-control-sm">*

*<option value="">Семестр</option>*

*@for($k = 1; $k <= 8; $k++)*

*<option value="{{ $k }}" {{ $k == @$\_GET['sem'] ? 'selected' : '' }}>{{ $k }}</option>*

*@endfor*

*</select>*

*</div>*

*<div class="col-sm-2">*

*<label>&nbsp;</label><input type="submit" class="btn btn-sm btn-info d-block" value="Показать">*

*</div>*

*</div>*

*</form>*

*@if($journals)*

*<hr>*

*{{ $journals->appends(Request::except('page'))->links() }}*

*<table class="table table-hover">*

*<thead>*

*<tr>*

*<th>Группа</th>*

*<th>Предмет</th>*

*<th class="text-center">Учебный год</th>*

*<th class="text-center">Семестр</th>*

*<th>Преподаватель</th>*

*<th></th>*

*</tr>*

*</thead>*

*<tbody>*

*@foreach($journals as $key => $j)*

*<tr>*

*<td>{{ $j->group->codes[$j->kurs] }}</td>*

*<td>{{ $j->subject->name }}</td>*

*<td class="text-center">{{ $j->year.'-'.($j->year + 1) }}</td>*

*<td class="text-center">{{ $j->semestr }}</td>*

*<td>{{ $j->teacher->shortName }}</td>*

*<td class="text-right">*

*<a href="/journal/{{$j->id}}" class="btn btn-sm btn-outline-info">*

*Перейти*

*</a>*

*</td>*

*</tr>*

*@endforeach*

*</tbody>*

*</table>*

*{{ $journals->appends(Request::except('page'))->links() }}*

*@endif*

*@endsection*

Для создания клиентской логики в преставлениях использовался как «чистый» JS, так и фреймворк VueJS. Он удобен для управления состоянием приложения, взаимодействия с API и описания более сложной логики. Пример JS-кода чата приведен в листинге 5.

Листинг 5 – код чата.

*const app = new Vue({*

*el: '#app',*

*data: {*

*hasMore: false,*

*topic: null,*

*user: null,*

*messages: [],*

*message: '',*

*file: '',*

*reply: null*

*},*

*methods: {*

*send: function() {*

*if(this.message || this.file) {*

*var formData = new FormData()*

*formData.append('text', this.message)*

*formData.append('topic\_id', this.topic.id)*

*if(this.reply) {*

*formData.append('reply\_id', this.reply.id)*

*}*

*if(this.file) {*

*formData.append('file', this.file)*

*}*

*var options = this.file ? {*

*headers: {'Content-Type': 'multipart/form-data'}*

*} : null*

*axios.post('/api/forum?api\_token=' + this.user.api\_token, formData, options)*

*.then(response => {*

*this.messages.push(response.data)*

*this.message = ''*

*this.file = ''*

*this.reply = null*

*this.$refs.room.scrollTo(0, this.$refs.room.scrollHeight \* 2)*

*})*

*}*

*},*

*load: function() {*

*return new Promise((resolve, reject) => {*

*axios.get('/api/forum/' + this.topic.id +*

*'?api\_token=' + this.user.api\_token +*

*'&skip=' + this.messages.length)*

*.then(response => {*

*this.messages = response.data.reverse().concat(this.messages)*

*this.hasMore = response.data.length >= 10*

*resolve()*

*})*

*})*

*},*

*setFile: function() {*

*this.file = this.$refs.file.files[0]*

*}*

*},*

*created() {*

*this.topic = <?=json\_encode($topic)?>;*

*this.user = <?=json\_encode(\Auth::user())?>;*

*this.load().then(() => this.$refs.room.scrollTo(0, this.$refs.room.scrollHeight))*

*}*

*});*

Также в проекте Laravel есть возможность создавать консольные команды и настраивать их выполнение по расписанию.

**3 ЭКОНОМИЧЕСКАЯ ЧАСТЬ**

**3.1 Расчет времени на создание программного продукта**

Для оценки трудоемкости, себестоимости и плана-графика для проектов по разработке программного обеспечения все расчеты экономических показателей производятся в соответствии с алгоритмической моделью оценки стоимости разработки программного обеспечения COCOMO II, разработанной Барри Боэмом.

Расчет времени на создание программного продукта (затраты рабочего времени на производство единицы продукции), измеряющийся в человеко-часах, проходит восемь этапов. Структура общего времени для создания программного продукта представлена в таблице 1.

Таблица 1 – Структура общего времени для создания программного продукта

|  |  |  |
| --- | --- | --- |
| № этапа | Обозначение времени  данного этапа | Содержание этапа |
| 1 | Тпо | Подготовка описания задачи |
| 2 | То | Описание задачи |
| 3 | Та | Разработка алгоритма |
| 4 | Тбс | Разработка блок-схемы алгоритма |
| 5 | Тн | Написание программы на языках PHP, JS |
| 6 | Тп | Набивка программы |
| 7 | Тот | Отладка и тестирование программы |
| 8 | Тд | Оформление документации, инструкции пользователя, пояснительной записки |

Время рассчитывается в человеко-часах, причем Тпо берется по фактически отработанному времени, а время остальных этапов определяется расчетом по условному числу команд Q.

Условное число команд Q определяется по формуле:

 (1)

 команд,

где q – коэффициент, учитывающий условное число команд в зависимости от типа задачи;

с – коэффициент, учитывающий новизну и сложность программы.

Выбрать значение коэффициента q можно из таблицы 2.

Программные продукты по степени новизны могут быть отнесены к одной из 4-х групп:

группа А – разработка принципиально новых задач;

группа Б – разработка оригинальных программ;

группа В – разработка программы с использованием типовых решений;

группа Г – разовая типовая задача.

Таблица 2 – Значение коэффициента q

|  |  |
| --- | --- |
| Тип задачи | Пределы изменений коэффициента |
| Задачи учета | от 1400 до 1500 |
| Задачи оперативного управления | от 1500 до 1700 |
| Задачи планирования | от 3000 до 3500 |
| Многовариантные задачи | от 4500 до 5000 |
| Комплексные задачи | от 5000 до 5500 |

По степени сложности программные продукты могут быть отнесены к одной из 3-х групп:

1 – алгоритмы оптимизации и моделирования систем;

2 – задачи учета, отчетности и статистики;

3 – стандартные алгоритмы.

Коэффициент c определяется из таблицы 3 на пересечении групп сложности и степени новизны.

Таблица 3 – Коэффициент с

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Язык  программирования | Группа сложности | Степень новизны | | | |
| А | Б | В | Г |
| Высокого уровня | 1 | 1,38 | 1,26 | 1,15 | 0,69 |
| 2 | 1,30 | 1,19 | 1,08 | 0,65 |
| 3 | 1,20 | 1,10 | 1,00 | 0,60 |
| Низкого уровня | 1 | 1,58 | 1,45 | 1,32 | 0,79 |
| 2 | 1,49 | 1,37 | 1,25 | 0,74 |
| 3 | 1,38 | 1,26 | 1,15 | 0,69 |

Согласно формуле 1, определяется условное число команд Q.

Вычисляется время, затраченное на каждый этап создания программного продукта:

1. Тпо (время на подготовку описания задачи), берется по факту и измеряется в человеко-часах и равно 16 часам.

2. То (время на описание задачи) определяется по формуле:

 (2)

(часов),

где *В* – коэффициент учета изменений задачи, который зависит от сложности задачи и числа изменений и выбирается в интервале от 1,2 до 1,5.

*К* – коэффициент, учитывающий квалификацию программиста (Таблица 4).

3. Та (время на разработку алгоритма) рассчитывается по формуле:

 (3)

Та=6300/(50\*0,8) = 158 (час).

Таблица 4 - Значение коэффициента К

|  |  |
| --- | --- |
| Стаж программиста | Значение коэффициента К |
| до 2-х лет | 0,8 |
| от 2 до 3 лет | 1,0 |
| от 3 до 5 лет | 1,1 – 1,2 |
| от 5 до 10 лет | 1,2 – 1,3 |
| свыше 10 лет | 1,3 – 1,5 |

4. Тбс (время на разработку блок-схемы) определяется аналогично Та по формуле 3.

Тбс=6300/(50\*0,8) = 158 (час).

5. Тн (время написания программы на языке программирования) определяется по формуле:

 (4)

 (час).

6. Тп (время набивки программы) определяется по формуле:

 (5)

 (часов)

7. Тот (время отладки и тестирования программы) определяется по формуле:

 (6)

 (часа).

8. Тд (Оформление документации, инструкции пользователя, пояснительной записки) берется по факту, измеряется в человеко-часах и равно 2 часам.

Далее производится подсчет общего времени на создание программного продукта:

+Тд

(7)

Т = 16+232+158+158+236+126+662+2 = 1590‬ (часов).

Для определения заработной платы, за выполненные проектные работы, необходимо знать затраты труда на разработку программы и квалификационный состав работников, выполняющих эти работы.

Согласно закону Республики Казахстан от 04 декабря 2019 года № 276-VІ ЗРК «О республиканском бюджете на 2020-2022 годы» статья 7 минимальный размер заработной платы с 01 января 2020 года составляет 42 500 тенге. Её размер может быть взят за основу при расчете заработной платы работников предприятий и организаций различных форм собственности, если иное не предусматривается действующим законодательством РК.

Согласно приказу Министра образования и науки Республики Казахстан от 29 января 2016 года № 123 «Об утверждении Реестра должностей гражданских служащих в сфере образования и науки» должность программиста относится к блоку «С» (административный персонал). Специалисты среднего уровня квалификации относятся к звену С3.

Заработная плата вышеуказанных работников рассчитывается в соответствии с Постановлением Правительства Республики Казахстан от 31 декабря 2015 года № 1193 «О системе оплаты труда гражданских служащих, работников организаций, содержащихся за счет средств государственного бюджета, работников казенных предприятий» (далее Постановление).

ДО (тарифная ставка) работников организаций определяется путем умножения соответствующих коэффициентов, утвержденных для исчисления их ДО (тарифной ставки) в зависимости от отнесения занимаемых должностей к функциональным блокам и стажа работы по специальности, присвоенных квалификационных разрядов (для рабочих), на размер БДО, установленного подпунктом 1) части первой пункта 1 Постановления. Для гражданских служащих, работников организаций, содержащихся за счет средств государственного бюджета, работников казенных предприятий размер базового должностного оклада (БДО) равен 17 697,00 тенге.

В таблице 5 представлена выдержка из Приложения 2 Постановления, касательно определения коэффициентов для исчисления должностных окладов.

Таблица 5- Коэффициенты для исчисления должностных окладов

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Звено | | Ступень | Стаж работы по специальности, в годах | | | | | | | | | | | | |
| 0-1 | 1-2 | | 2-3 | 3-5 | 5-7 | 7-10 | 10-13 | 13- 16 | | 16-20 | 20- 25 | Свыше 25 |
| Блок С – административный персонал | | | | | | | | | | | | | | | |
| С3 |  | | 3,31 | 3,35 | 3,39 | | 3,43 | 3,46 | 3,50 | 3,54 | 3,57 | 3,61 | | 3,65 | 3,68 |

1. Месячная тарифная ставка работника определяется:

 (8)

Тм=17 697,00 \* 3,31 = 58 577,07 (тенге),

где: БДО – базовый должностной оклад, 17 697,00 тенге;

Kt – тарифный коэффициент (Таблица 5).

2. Дневная тарифная ставка Тд при среднемесячном балансе рабочего времени 20,50 рабочих дня в месяце определяется:

Тд = Тм / 20,50 (9)

Тд= 58 577,07 / 20,50 = 2 857,42 (тенге)

3. Часовая тарифная ставка Тч для данного разряда при 8-часовом рабочем дне рассчитывается по формуле:

 (10)

Тч =2 857,42 / 8 = 357,18 (тенге)

Далее определяется заработная плата работника:

 (11)

Зп = 357,18 \* 1 590‬= 567 916,20‬ (тенге),

где Т – общее время на создание программного продукта.

Далее рассчитываются отчисления от заработной платы работника. В соответствии со статьей 485 п.1 Кодекса Республики Казахстан «О налогах и других обязательных платежах в бюджет» (по состоянию на 01.01.2020 г.) заработная плата работника (база обложения) является объектом обложения социальным налогом по ставке 9,5%:

СНиСО = (Зп - ОПВ(10%))\*9,5%, (12)

где СНиСО - социальный налог и социальные отчисления,

ОПВ (10%) = Зп \* 10% (13)

ОПВ (10%) = 567 916,2‬\*10% = 56 791,62‬ (тенге)

СНиСО = (567 916,2 – 56 791,62‬) \* 9,5% = 48 556,84 (тенге).

Размер отчислений на обязательное социальное медицинское страхование (ООСМС) с 1 января 2020 года составляет 2,0 % от объекта обложения, также обязательно округление до целых величин по арифметическим правилам:

ООСМС (2,0 %) = Зп \* 2,0 % (14)

ООСМС (2%) = 567 916,2\*2,0 % = 11 358,00 (тенге).

**3.2 Расчет себестоимости 1-го машино-часа работы компьютере**

Основой для расчета расходов на содержание и эксплуатацию ПЭВМ, относящихся к данному программному продукту, является себестоимость 1-го машино-часа работы ПЭВМ.

Сначала определяются годовые затраты каждого компонента себестоимости (Таблица 6), в число которых входят:

Таблица 6 - Параметры технического средства

|  |  |
| --- | --- |
| Наименование комплектующих и периферии | Стоимость (Тенге) |
| Ноутбук Acer Aspire 3 A315-21-65LJ | 167 000,00 |
| Мышь | 1 500,00 |
| Итого: | 168 500,00 |

Амортизационные отчисления определяются в размере от 20 % до 40% от балансовой стоимости ПЭВМ.

А = Кб \* 20 % (15)

где Кб – балансовая стоимость одной ПЭВМ с периферией.

 тенге

Работа над данной программой заняла 1590 часов, учитывая, что по законодательству рабочий день составляет 8 часов, а в месяце в среднем 20,50 рабочих дня, то берем амортизацию за 9,7 месяца, поэтому в расчеты включаем амортизацию в сумме 18 282,25 тенге.

Аи=А/12\*9,7 (16)

Аи = 33700,00 / 12 \* 9,7 = 27 240,83 тенге

2. Затраты на электроэнергию складываются из:

- затраты на силовую электроэнергию;

- затраты на электроэнергию, которая расходуется на освещение.

Затраты на силовую электроэнергию определяются по формуле:

 (17)

где: Цэ – стоимость 1кВт/часа в тенге; Р – суммарная мощность ПЭВМ с периферией в кВт/часах. Стоимость 1 кВт/часа составляет 16,96 тенге.

Р = 0,5 кВт/час.

Зс.эн=1590\*16,96\*0,5 = 13 483,20‬‬ (тенге)

Затраты на электроэнергию, которая идет на освещение определяется по формуле:

 (18)

Зосв=1590\*16,96\*0,22= 5 932,61 (тенге)

где: Росв – суммарная мощность, которая идет на освещение в кВт/часах. В дипломном проекте принимаем Росв = 0,22 кВт/час.

Теперь определяем общие затраты на электроэнергию:

 (19)

Зэн = 13 483,2 + 5 932,6 = 19 415,81 (тенге).

3. Затраты на расходные материалы Мр берутся по факту.

В их число входят диски, картриджи и бумага для принтеров и т.д. При разработке данного программного продукта затрат на расходные материалы не имели места.

 (тенге) (20)

4. Расходы на профилактику Рпр составляют 2% от балансовой стоимости ПЭВМ с периферией.

 (21)

 (тенге)

5. Прочие производственные расходы берутся в размере 15% от вышеперечисленных затрат, обеспечивающих функционирование ПЭВМ.

 (22)

Рпроч = (27 240,83 + 19 415,81 + 0,00 + 3370,00) \* 15% = 7 504,00 (тенге)

6. Расходы на содержание и эксплуатацию (Рсэ) 1-ой ПЭВМ.

Все вышеперечисленные компоненты суммируются, определяются годовые расходы на содержание и эксплуатацию ПК (Рсэ):

Рсэ = Аи+Зэ+Мр+Рпроч (23)

Рсэ = 27 240,83 +19 415,81 +0,00+7 504,00 = 54 160,64 (тенге).

Себестоимость одного машино-часа работы ПК рассчитывается по формуле:

Смч = Рсэ/Фэф (24)

где Фэф – годовой фонд рабочего времени: 1968=246 дней \* 8 часов в день.

Смч = 54 160,64/1968 = 27,52 (тенге).

**3.3 Расчёт расходов на содержание и эксплуатацию ПК**

Теперь зная себестоимость 1-го Машино-часа работы ПК и время на создание программного продукта, которое требовало использование ПК, можно определить расходы на содержание и эксплуатацию ПК, относящихся к данному программному продукту (Рсэп):

 (25)

где: Смч – себестоимость одного машино-часа работы ПК; Т – суммарное время этапов разработки программного продукта. Рсэп = 27,52\*1590 = 43 756,80‬ (тенге).

**3.4 Расчет себестоимости программного продукта**

В себестоимости программного продукта входят следующие элементы:

1. заработная плата Зп исполнителя работ по созданию программного продукта;

2. социальный налог и социальные отчисления, отчисления на обязательное социальное медицинское страхование;

3. амортизационные отчисления;

4. расходы на содержание и эксплуатацию ПЭВМ, относящихся к данному программному продукту.

Теперь, сложив все элементы, можно определить себестоимость программного продукта:

Сп.п = Зп + СНиСО+ОСМС+Аи+Зэн+Мр+Рпр (26)

Сп.п = 567 916,20 + 48 556,84 + 11 358,00 + 27 240,83 + 19 415,81 + 0,00 +

+ 3370,00 = 677 857,68‬ (тенге).

Сводная таблица технико-экономических показателей. Результаты расчетов экономической части дипломной работы сводится в итоговую Таблицу 7.

Таблица 7 - Технико-экономические показатели

|  |  |  |
| --- | --- | --- |
| № | Наименование показателя | Сумма, тенге |
| 1 | Заработная плата исполнителя | 567 916,20 |
| 2 | Социальный налог и социальные отчисления | 48 556,84 |
| 3 | Отчисления на обязательное социальное медицинское страхование | 11 358,00 |
| 4 | Амортизационные отчисления | 27 240,83 |
| 5 | Затраты на электроэнергию | 19 415,81 |
| 6 | Затраты на расходные материалы | 0,00 |
| 7 | Расходы на профилактику | 3370,00 |
| 8 | Себестоимость программного продукта | 677 857,68 |

Себестоимость программного продукта составляет 677 857,68‬ тенге.

**3.5 Расчет срока окупаемости**

Окупаемость программного продукта является актуальным вопросом для любого проекта.

Проект окупит себя в тот момент, когда прибыль, полученная от всех сделок, совершённых при помощи проекта, сравняется с суммой затрат на его создание.

Оценить срок окупаемости проекта можно по формуле:

So=Сп.п./(Kd\*P), (27)

где So - срок окупаемости, дни;

Сп.п – себестоимость программного продукта;

Kd – общее количество сделок; P – прибыль.

Проведем примерный расчет срока окупаемости проекта.

Предположим:

- затраты на проект 677 857,68‬ тенге, согласно вышеприведенному расчету себестоимости программного продукта;

- примерная прибыль с каждой сделки 50 000 тенге.

Ориентировочное число покупателей (сделок) в день 1-2.

Срок окупаемости = (677 857,68‬) / (от 1 до 2) \* 50 000) = 7 – 14 (дней).

Средняя прибыль в месяц = (от 1 до 2) \* 50 000 \* 30 = 1 500 000‬– 3 000 000 (тенге).

**4 МЕРОПРИЯТИЯ ПО ТЕХНИКЕ БЕЗОПАСНОСТИ И ОХРАНЕ ОКРУЖАЮЩЕЙ СРЕДЫ**

**4.1 Общие принципы и законодательство по охране труда**

В соответствие с Законом Республики Казахстан «О безопасности и охране труда», охрана труда представляет собой действующую на основании соответствующих законодательных и иных нормативных актов систему социально - экономических, организационных, технических, гигиенических и лечебно- профилактических мероприятий и средств, обеспечивающих безопасность, сохранение здоровья и работоспособности человека в процессе труда. Право на охрану труда имеют все работники, состоящие в трудовых отношениях с предприятиями, учреждениями, кооперативами, фермерскими хозяйствами и другими организациями различных форм собственности и хозяйствования, в том числе с отдельными нанимателями; члены кооперативов; студенты высших учебных заведений и учащиеся средних специальных учебных заведений и общеобразовательных школ, проходящие производственную практику и производственное обучение; военнослужащие, привлеченные к работам, не связанным с несением воинской службы; лица, отбывающие наказание по приговору суда, в период их работы на предприятиях, определяемых органами, ведающими исполнением приговоров, а также участники любых других видов трудовой деятельности, организуемой в интересах общества и государства.

Национальная политика в области охраны труда предусматривает единство действий органов государственной власти и управления всех уровней при участии профсоюзов и работодателей.

**4.2 Пожарная безопасность и электро-безопасность**

Пожар – это неконтролируемое горение вне отведенного для этого очага. Пожары представляют большую опасность для работающих и могут причинить огромный материальный ущерб.

Пожар на производстве может возникнуть вследствие причин неэлектрического и электрического характера.

К причинам неэлектрического характера относятся:

* неисправность производственного оборудования и нарушение технологического процесса;
* халатное и неосторожное обращение с огнем (курение, оставление без присмотра нагревательных приборов);
* неправильное устройство и неисправность вентиляционной системы;
* самовоспламенение или самовозгорание веществ.

К причинам электрического характера относятся:

* короткое замыкание;
* перегрузка проводов;
* большое переходное сопротивление;
* искрение;
* статическое электричество.

Пожарная безопасность может быть обеспечена мерами пожарной профилактики и активной пожарной защитой. Понятие о пожарной профилактике включает комплекс мероприятий, необходимых для предупреждения возникновения пожара или уменьшения его последствий. Под активной пожарной защитой понимаются меры, обеспечивающие успешную борьбу с возникающими пожарами или взрывоопасными ситуациями.

Профилактические методы борьбы с пожарами на участке ПЭВМ предусматривают:

* организационные, правильное содержание помещений, противопожарный инструктаж служащих, издание приказов по вопросам усиления пожарной безопасности и т.д.;
* технические, соблюдение противопожарных правил, норм при проектировании помещений, при устройстве электропроводов и оборудования, отопления, вентиляции, освещения;
* режимные, запрещение курения в не установленных местах, производство пожароопасных работ в помещении машинного зала ВЦ и т.д.;
* эксплуатационные, своевременные профилактические осмотры, ремонты оборудования.

При проектировании и строительстве зданий и помещений (в частности, машинного зала) также должны быть соблюдены противопожарные меры:

* защита деревянных конструкций достигается пропиткой огнезащитными химическими препаратами (например, антипиринами), покрытием огнезащитными красками;
* для ограничения распространения пожара устраивают противопожарные преграды стены, перегородки, перекрытия, двери, ворота, люки, окна. Все это должно быть выполнено из несгораемых материалов.

Необходимо предусмотреть безопасную эвакуацию людей на случай возникновения пожара. При пожаре люди должны покинуть помещение в течение минимального времени. Число эвакуационных выходов из зданий, помещений должно составлять не менее двух.

Активные методы борьбы с пожарами на участке ПЭВМ:

* изоляция очага горения от воздуха с помощью твердых веществ;
* охлаждение очага горения ниже определенных температур. Достигается с помощью воды, но у нее есть ограничения на тушение легковоспламеняющихся веществ, электроустановок и др., поэтому чаще применяют углекислый газ, который при соединении с атмосферой снижает температуру до -78º С;
* интенсивное торможение скорости химической реакции в пламени. В основном для этой цели применяют порошки;
* оптимальная направленность излучаемого осветительными приборами светового потока;
* механический срыв пламени в результате воздействия на него сильной струи газа или воды.

Для тушения пожаров в машинном зале ВЦ необходимо применять углекислотные и порошковые огнетушители, которые обладают высокой скоростью тушения, большим временем действия, возможностью тушения электроустановок, высокой эффективностью борьбы с огнем.

Исходя из норм пожарной безопасности для машинного зала ВЦ площадью до 100 м2 требуются следующие первичные средства пожаротушения:

* один углекислотный огнетушитель типа ОУ-5 или ОУ-8, с помощью которого можно тушить загорания различных материалов и установок напряжением до 1000 В;
* один химпенный (ОХП-10) или воздушно-пенный огнетушитель (ОВП-5 или ОВП-10), с помощью которого можно тушить твердые материалы и горючие жидкости (кроме установок под напряжением);
* войлок, кошму или асбест (11; 21,5; 22 м).

Помещение машинного зала должно быть оборудовано пожарными извещателями, которые позволяют оповестить дежурный персонал о пожаре.

Пожарные извещатели преобразуют неэлектрические физические величины (излучение тепловой или световой энергии, движение частиц дыма) в электрические, которые в виде сигнала определенной формы направляются по проводам на приемную станцию.

В качестве пожарных извещателей в машинном зале устанавливаются дымовые фотоэлектрические извещатели типа ИДФ-1 или ДИП-1.

В целях обеспечения безопасности обслуживающего персонала и обеспечения наиболее быстрого устранения ситуаций, угрожающих здоровью либо жизни людей, на объекте предусмотрен комплекс мер по предупреждению и наиболее быстрому устранению подобных ситуаций, а также, наличие средств защиты и пожаротушения. К ним относятся огнетушители и прочие противопожарные принадлежности, а также, системы пожаротушения, сигнализация и системы оповещения персонала.

Работа на персональных компьютерах связано со значительной умственной, эмоциональной, зрительной и статической нагрузками которые постоянно испытывает пользователь, работая с компьютерами. Согласно типовой инструкции Государственной инспекции безопасности труда необходимо соблюдать следующие требования безопасности:

* помещение, в котором находится компьютер, должно быть светлым, сухим, теплым. Для полов должно использоваться покрытие поливинилхлоридное, двухслойное, антистатическое, а для отделки дверей и мебели поливинилхлоридный антистатический материал;
* запылённость воздуха не должна превышать 0.75 мг/м3. На одного инженера - программиста должен приходиться объём помещения 15м3 при площади 4.5 м2 (без учёта проходов и оборудования). В течение трудового дня необходимо обеспечить воздухообмен помещения объёмом от 25 до 50 м3, отвод влаги в диапазоне от 350 до 500 г и тепла 50 кДж на каждый килограмм массы тела работающего;
* плотность магнитного потока в 50 сантиметрах вокруг дисплея составляет около 250 нТл в диапазоне частот от 5 Гц до 2КГц; поверхностный электростатический потенциал составляет 500 В. Время работы за дисплеем не должно превышать четырех часов в сутки;
* конструкция компьютера должна обеспечивать надежную защиту от ионизирующих и неионизирующих излучений. Рентгеновское излучение от дисплея не должно превышать 100 мкР/ч на расстоянии 5 см от экрана и каждой из сторон корпуса, напряженность электростатического поля не должна превышать 15 кВт/м;
* радиаторы и трубопроводы отопительной водопроводной систем оборудуются диэлектрическими (деревянными и т.п.) ограждениями;
* при работе на ПК имеется возможность прикосновения к корпусам электрооборудования, которые могут оказаться под напряжением. Необходимо устройство защитного заземления. Защитному заземлению подлежат металлические части электрооборудования, доступные для прикосновения человека. Заземляемое оборудование должно быть присоединено к заземляющей магистрали посредством отдельного заземленного проводника;
* запрещается последовательное включение в заземляющий проводник нескольких устройств. Необходимо 1 раз в год осуществлять замеры неиспользованного рентгеновского излучения, напряженность электростатического поля на рабочем месте, величину сопротивления контуров заземления и изоляции электропроводки. Кроме того, необходимо обеспечить надежное заземление. Эффективным заземлением является заземление трубчатого типа с толщиной стенки 3,5 мм. Длина трубы обычно составляет 250 см, диаметр 5 см. Заземлители располагаются по четырехугольному контуру, с глубиной заложения около 80 см.

По технологии изготовления фильтры бывают сеточные, пленочные и стеклянные. Фильтры могут крепиться к передней стенке монитора, навешиваться на его верхнюю кромку, вставляться в специальный желобок вокруг экрана или надеваться на монитор.

Сеточные фильтры практически не защищают от электромагнитного излучения. Однако они неплохо ослабляют блики от внешнего освещения, что пир интенсивной работе за компьютером является немаловажным фактором.

Пленочные фильтры также не защищают от статического электричества, но значительно повышают контрастность изображения, практически полностью поглощают ультрафиолетовое излучение и снижают уровень рентгеновского излучения.

Что касается стеклянных фильтров, то они выпускаются в нескольких различных модификациях. Простые стеклянные фильтры снимают статический заряд, ослабляют низкочастотные электромагнитные поля, снижают интенсивность ультрафиолетового излучения и повышают контрастность изображения.

Эффективным средством профилактики несчастных случаев является наиболее удачное расположение оборудования, использование, по возможности, приборы и оборудования с наиболее оптимальными конструктивными решениями. Важным средством обеспечения безопасности служит надежная изоляция токонесущих частей, кабелей, а также, заземление корпусов всех приборов и металлических частей оборудования.

**4.3 Санитарно-гигиенические требования к помещениям**

**4.3.1 Требования к освещению помещений и рабочих мест**

На рабочем месте, в котором размещено автоматизированное рабочее место пользователя должно быть естественное и искусственное освещение. Естественное освещение обеспечивается через оконные проемы с коэффициентом естественного освещения КЕО не ниже 1,2 % в зонах с устойчивым снежным покровом и не ниже 1,5% на остальной территории. Световой поток из оконного проема должен падать на рабочее место оператора с левой стороны.

Искусственное освещение в помещениях эксплуатации компьютеров должно осуществляться системой общего равномерного освещения.

Освещенность на поверхности стола в зоне размещения документа должна быть 300-500 лк. Допускается установка светильников местного освещения для подсветки документов. Местное освещение не должно создавать бликов на поверхности экрана и увеличивать освещенность экрана более 300 лк. Прямую блескость от источников освещения следует ограничить. Яркость светящихся поверхностей (окна, светильники), находящихся в поле зрения, должна быть не более 200 кд/м2.

Отраженная блескость на рабочих поверхностях ограничивается за счет правильного выбора светильника и расположения рабочих мест по отношению к естественному источнику света. Яркость бликов на экране монитора не должна превышать 40 кд/м2. Показатель ослепленности для источников общего искусственного освещения в помещениях должен быть не более 20, показатель дискомфорта в административно-общественных помещениях не более 40. Соотношение яркости между рабочими поверхностями не должно превышать 3:1 — 5:1, а между рабочими поверхностями и поверхностями стен и оборудования 10:1.

Для искусственного освещения помещений с персональными компьютерами следует применять светильники типа ЛПО36 с зеркализованными решетками, укомплектованные высокочастотными пускорегулирующими аппаратами. Допускается применять светильники прямого света, преимущественно отраженного света типа ЛПО13, ЛПО5, ЛСО4, ЛПО34, ЛПО31 с люминесцентными лампами типа ЛБ. Допускается применение светильников местного освещения с лампами накаливания. Светильники должны располагаться в виде сплошных или прерывистых линий сбоку от рабочих мест параллельно линии зрения пользователя при разном расположении компьютеров. При периметральном расположении — линии светильников должны располагаться локализовано над рабочим столом ближе к его переднему краю, обращенному к оператору. Защитный угол светильников должен быть не менее 40 градусов. Светильники местного освещения должны иметь не просвечивающийся отражатель с защитным углом не менее 40 градусов.

Для обеспечения нормативных значений освещенности в помещениях следует проводить чистку стекол оконных проемов и светильников не реже двух раз в год и проводить своевременную замену перегоревших ламп.

## 

## 4.3.2 Требования к организации и оборудованию рабочих мест

Рабочие места с персональными компьютерами по отношению к световым проемам должны располагаться так, чтобы естественный свет падал сбоку, желательно слева.

Схемы размещения рабочих мест с персональными компьютерами должны учитывать расстояния между рабочими столами с мониторами: расстояние между боковыми поверхностями мониторов не менее 1,2 м, а расстояние между экраном монитора и тыльной частью другого монитора не менее 2,0 м.

Рабочий стол может быть любой конструкции, отвечающей современным требованиям эргономики и позволяющей удобно разместить на рабочей поверхности оборудование с учетом его количества, размеров и характера выполняемой работы. Целесообразно применение столов, имеющих отдельную от основной столешницы специальную рабочую поверхность для размещения клавиатуры. Используются рабочие столы с регулируемой и нерегулируемой высотой рабочей поверхности. При отсутствии регулировки высота стола должна быть в пределах от 680 до 800 мм.

Глубина рабочей поверхности стола должна составлять 800 мм (допускаемая не менее 600 мм), ширина — соответственно 1 600 мм и 1 200 мм. Рабочая поверхность стола не должна иметь острых углов и краев, иметь матовую или полуматовую фактору.

Рабочий стол должен иметь пространство для ног высотой не менее 600 мм, шириной — не менее 500 мм, глубиной на уровне колен — не менее 450 мм и на уровне вытянутых ног — не менее 650 мм.

Быстрое и точное считывание информации обеспечивается при расположении плоскости экрана ниже уровня глаз пользователя, предпочтительно перпендикулярно к нормальной линии взгляда (нормальная линия взгляда 15 градусов вниз от горизонтали).

Клавиатура должна располагаться на поверхности стола на расстоянии 100-300 мм от края, обращенного к пользователю.

Для удобства считывания информации с документов применяются подвижные подставки (пюпитры), размеры которых по длине и ширине соответствуют размерам устанавливаемых на них документов. Пюпитр размещается в одной плоскости и на одной высоте с экраном.

Для обеспечения физиологически рациональной рабочей позы, создания условий для ее изменения в течение рабочего дня применяются подъемно-поворотные рабочие стулья с сиденьем и спинкой, регулируемыми по высоте и углам наклона, а также расстоянию спинки от переднего края сидения.

Конструкция стула должна обеспечивать:

1 Ширину и глубину поверхности сиденья не менее 400 мм;

2 Поверхность сиденья с закругленным передним краем;

3 Регулировку высоты поверхности сиденья в пределах 400-550 мм и углом наклона вперед до 15 градусов и назад до 5 градусов.;

4 Высоту опорной поверхности спинки 300±20 мм, ширину — не менее 380 мм и радиус кривизны горизонтальной плоскости 400 мм;

5 Угол наклона спинки в вертикальной плоскости в пределах 0±30 градусов;

6 Регулировку расстояния спинки от переднего края сидения в пределах 260-400 мм;

7 Стационарные или съемные подлокотники длиной не менее 250 мм и шириной 50-70 мм;

8 Регулировку подлокотников по высоте над сиденьем в пределах 230±30 мм и внутреннего расстояния между подлокотниками в пределах 350-500 мм;

9 Поверхность сиденья, спинки и подлокотников должна быть полумягкой, с нескользящим не электризующимся, воздухонепроницаемым покрытием, легко очищаемым от загрязнения.

Рабочее место должно быть оборудовано подставкой для ног, имеющей ширину не менее 300 мм, глубину не менее 400 мм, регулировку по высоте в пределах до 150 мм и по углу наклона опорной поверхности подставки до 20 град. Поверхность подставки должна быть рифленой и иметь по переднему краю бортик высотой 10 мм.

## 4.3.3 Требования к шуму и вибрации в помещениях

Уровни шума на рабочих местах пользователей персональных компьютеров не должны превышать значений, установленных СанПиН и составляют не более 50 дБА. На рабочих местах в помещениях для размещения шумных агрегатов уровень шума не должен превышать 75 дБА, а уровень вибрации в помещениях допустимых значений по СН 2.2.4/2.1.8.566-96 категория 3, тип «в».

Снизить уровень шума в помещениях можно использованием звукопоглощающих материалов с максимальными коэффициентами звукопоглощения в области частот 63-8000 Гц для отделки стен и потолка помещений. Дополнительный звукопоглощающий эффект создают однотонные занавески из плотной ткани, повешенные в складку на расстоянии 15-20 см от ограждения. Ширина занавески должна быть в 2 раза больше ширины окна.

## 4.4 Режим труда и отдыха при работе с компьютером

Режим труда и отдыха предусматривает соблюдение определенной длительности непрерывной работы на ПК и перерывов, регламентированных с учетом продолжительности рабочей смены, видов и категории трудовой деятельности.

Виды трудовой деятельности на ПК разделяются на 3 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.

Если в течение рабочей смены пользователь выполняет разные виды работ, то его деятельность относят к той группе работ, на выполнение которой тратится не менее 50% времени рабочей смены.

Категории тяжести и напряженности работы на ПК определяются уровнем нагрузки за рабочую смену: для группы А — по суммарному числу считываемых знаков; для группы Б — по суммарному числу считываемых или вводимых знаков; для группы В — по суммарному времени непосредственной работы на ПК. В таблице 9 приведены категории тяжести и напряженности работ в зависимости от уровня нагрузки за рабочую смену.

Количество и длительность регламентированных перерывов, их распределение в течение рабочей смены устанавливается в зависимости от категории работ на ПК и продолжительности рабочей смены.

Таблица 9 - Виды категорий трудовой деятельности с ПК

|  |  |  |  |
| --- | --- | --- | --- |
| Категория работы по тяжести и напряженности | Уровень нагрузки за рабочую смену при видах работы на ПК | | |
| Группа А  Количество  знаков | Группа Б  Количество знаков | Группа В  Время работы, ч |
| I | до 20000 | до 15000 | до 2,0 |
| II | до 40000 | до 30000 | до 4,0 |
| III | до 60000 | до 40000 | до 6,0 |

При 8-часовой рабочей смене и работе на ПК регламентированные перерывы следует устанавливать:

* для первой категории работ через 2 часа от начала смены и через 2 часа после обеденного перерыва продолжительностью 15 минут каждый;
* для второй категории работ — через 2 часа от начала рабочей смены и через 1,5-2,0 часа после обеденного перерыва продолжительностью 15 минут каждый или продолжительностью 10 минут через каждый час работы;
* для третьей категории работ — через 1,5- 2,0 часа от начала рабочей смены и через 1,5-2,0 часа после обеденного перерыва продолжительностью 20 минут каждый или продолжительностью 15 минут через каждый час работы.

При 12-часовой рабочей смене регламентированные перерывы должны устанавливаться в первые 8 часов работы аналогично перерывам при 8-часовой рабочей смене, а в течение последних 4 часов работы, независимо от категории и вида работ, каждый час продолжительностью 15 минут.

Продолжительность непрерывной работы на ПК без регламентированного перерыва не должна превышать 2 часа.

При работе на ПК в ночную смену продолжительность регламентированных перерывов увеличивается на 60 минут независимо от категории и вида трудовой деятельности.

Эффективными являются нерегламентированные перерывы (микро-паузы) длительностью 1-3 минуты.

Регламентированные перерывы и микропаузы целесообразно использовать для выполнения комплекса упражнений и гимнастики для глаз, пальцев рук, а также массажа. Комплексы упражнений целесообразно менять через 2-3 недели.

Пользователям ПК, выполняющим работу с высоким уровнем напряженности, показана психологическая разгрузка во время регламентированных перерывов и в конце рабочего дня в специально оборудованных помещениях (комнатах психологической разгрузки).

## 4.5 Медико-профилактические и оздоровительные мероприятия

Все профессиональные пользователи ПК должны проходить обязательные предварительные медицинские осмотры при поступлении на работу, периодические медицинские осмотры с обязательным участием терапевта, невропатолога и окулиста, а также проведением общего анализа крови и ЭКГ.

Не допускаются к работе на ПК женщины со времени установления беременности и в период кормления грудью.

Близорукость, дальнозоркость и другие нарушения рефракции должны быть полностью корригированы очками. Для работы должны использоваться очки, подобранные с учетом рабочего расстояния от глаз до экрана дисплея. При более серьезных нарушениях состояния зрения вопрос о возможности работы на ПК решается врачом-офтальмологом.

Досуг рекомендуется использовать для пассивного и активного отдыха (занятия на тренажерах, плавание, езда на велосипеде, бег, игра в теннис, футбол, лыжи, аэробика, прогулки по парку, лесу, экскурсии, прослушивание музыки и т.п.). Дважды в год (весной и поздней осенью) рекомендуется проводить курс витаминотерапии в течение месяца. Следует отказаться от курения. Категорически должно быть запрещено курение на рабочих местах.

**ЗАКЛЮЧЕНИЕ**

Во все времена система образования была и остается одной из важнейших сфер общества. От качества подготовки специалистов зависит качество жизни в будущем. Поэтому необходимо обеспечить максимально четкую работу сотрудников организаций образования. Преподаватели не должны тратить время и силы на рутинную работу и выполнение двойной работы, а посвятить себя непосредственно подготовке занятий и работе со студентами.

Сейчас все больше рабочих процессов подвергается автоматизации. Это позволяет сократить расходы времени и повысить точность и надежность выполняемых работ. Если выполняемая работа заключается в постоянной сверке связанных данных или их повторный ввод, целесообразно переложить эту задачу на автоматизированную систему.

Также информатизация образования дает возможность учиться удаленно, что делает получение знаний более доступным, удобным и современным.

В дипломном проекте был разработан веб-сервис, представляющий собой автоматизированную информационную систему колледжа, выполняющий следующие функции:

- формирование, редактирование и копирование учебных планов;

- редактор РП и автоматическая генерация КТП;

- автоматизированный редактор расписания с проверкой наложений;

- возможность вносить изменения в расписании на определенную дату;

- учет вычитки часов при внесении изменений в расписание и формирование тематического плана;

- синхронизация дат в журнале, расписании и КТП;

- электронный журнал и зачетка;

- просмотр инфографики по личному прогрессу;

- выгрузка Формы 2, Формы 3 и нагрузки преподавателя;

- авторизация по следующим ролям: администратор, заведующий, диспетчер, студент, преподаватель;

- разграничение прав доступа;

- автоматическая генерация учетных данных;

- форум для студентов и преподавателей.

Также был сделан акцент на интеграцию с MS Office. Поддерживается импорт учебного плана, справочников дисциплин, контингента и персонала из файла. Выходные документы скачиваются в форматах .docx и .xlsx.

Благодаря использованию различных элементов управления, применению современных технологий веб-разработки, удалось создать привлекательный, функциональный, удобный и интуитивно-понятный пользовательский интерфейс, с возможностью быстрой модернизации в случае необходимости. Все модули логически взаимосвязаны, что позволяет создавать новые надстройки и интегрировать в существующую систему.

Помимо самого интерфейса сайта, создано API для получения данных о расписании и для форума. Данное API также можно использовать как на других сайтах, так и в мобильных приложениях.

**В данном дипломном проекте были рассмотрены актуальные вопросы автоматизации учебного процесса, проанализирована целесообразность и создан сайт, реализующий необходимые функции.**

**В заключении дипломного проекта можно сказать, что в разработанном веб-приложении удачно сочетается интерфейс с его функциональностью и простотой использования. Интерфейс спроектирован таким образом, чтобы нужные ссылки и информация были на виду, а цвета, типографика и эффекты анимации способствовали удобству пользования и информативности.**

**Таким образом, материалы дипломного проекта показывают, что поставленные в нем задачи, сформулированные выше во введении, полностью решены.**

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

|  |  |
| --- | --- |
| 1 | Стаффер М. Laravel. Полное руководство. 2-е издание / М. Стаффер, пер. В. Сауль. - СПб: Питер, 2020. - 512 с. |
| 2 | Круг С. Веб-дизайн: книга Стива Круга или «не заставляйте меня думать!», 2-е издание. – Пер. с англ. – СПб: Символ-Плюс, 2008. – 224 с. |
| 3 | Сильвио Морето Bootstrap в примерах. / Пер. с англ. Рагимов Р. Н. / Науч. ред. Киселев А. Н. – М.: ДМК Пресс, 2017. – 314 с |
| 4 | Дронов В. А. Laravel. Быстрая разработка современных динамических Web-сайтов на PHP, MySQL, HTML и CSS. — СПб.: БХВ-Петербург, 2018. — 768 с. |
| 5 | Хэнчетт Э., Листуон Б. Vue.js в действии / Хэнчетт Э., Листуон Б. - Санкт-Петербург: Питер, 2019. - 304 с. |
| 6 | Д. Крокфорд. Как устроен JavaScript (How JavaScript Works) Д. Крокфорд. - Санкт-Петербург: Питер, 2019. - 304 с. |
| 7 | М. Хавербеке. Выразительный JavaScript (Eloquent JavaScript, 3rd Edition: A Modern Introduction to Programming) Д. Крокфорд. - Санкт-Петербург: Питер, 2019. - 480 с. |
| 8 | Котеров Д. PHP 7 в подлиннике / Д. Котеров, И. Симдянов. -СПб.: БХВ-Петербург, 2016. -1088 C. |
| 9 | Андерсон С.   Приманка для пользователей: создаем привлекательный сайт / С. Андерсон; [пер. с англ. С. Силинский]. - Москва: Питер, 2013. - 234 с. |
| 10 | Фрэйн Б. HTML5 и CSS3. Разработка сайтов для любых браузеров и устройств / Б. Фрэйн; [перевод с английского В. Черник]. - Санкт-Петербург [и др.]: Питер, 2014. - 298 с. |
| 11 | Шляхтина С. Джентльменский набор для быстрого создания сайта / С. Шляхтина // КомпьютерПресс. - 2007. - № 2. - С. 83-88. |
| 12 | Хворостьянова С. В. Веб-сайт: требования к информационной структуре и наполнению / С. В. Хворостьянова // Современная библиотека. - 2011. - № 1. - С. 68-73. |

**ПРИЛОЖЕНИЕ А**

**ЛИСТИНГ САЙТА**

Файл app/Plan.php

*<?php*

*namespace App;*

*use Illuminate\Database\Eloquent\Model;*

*use App\Lesson;*

*use App\Rating;*

*use App\Result;*

*class Plan extends Model*

*{*

*public static function boot()*

*{*

*parent::boot();*

*static::created(function ($plan) {*

*$plan->generateLessons();*

*});*

*static::updated(function ($plan) {*

*$plan->generateLessons();*

*});*

*}*

*public $timestamps = false;*

*protected $fillable = ['group\_id', 'semestr', 'cikl\_id', 'subject\_id', 'theory', 'practice', 'lab', 'project', 'is\_exam', 'is\_project', 'is\_zachet', 'controls', 'total', 'weeks', 'subgroup', 'shifr', 'shifr\_kz', 'year', 'teacher\_id', 'exam', 'consul'];*

*public function subject()*

*{*

*return $this->belongsTo('App\Subject');*

*}*

*public function cikl()*

*{*

*return $this->belongsTo('App\Cikl');*

*}*

*public function teacher()*

*{*

*return $this->belongsTo('App\Teacher')->withDefault();*

*}*

*public function group()*

*{*

*return $this->belongsTo('App\Group')->withDefault();*

*}*

*public function lessons()*

*{*

*return $this->hasMany('App\Lesson')->with('part')->orderBy('order', 'asc');*

*}*

*public function graphicExam()*

*{*

*return $this->hasOne('App\Exam')->withDefault();*

*}*

*public function getKursAttribute()*

*{*

*return $this->year - $this->group->year\_create + 1;*

*}*

*public function getProjectMainAttribute()*

*{*

*return static::where('group\_id', $this->group\_id)*

*->where('cikl\_id', $this->cikl\_id)*

*->where('subject\_id', $this->subject\_id)*

*->where('subgroup', '<>', 2)*

*->sum('project');*

*}*

*public function getPracticeMainAttribute()*

*{*

*return static::where('group\_id', $this->group\_id)*

*->where('cikl\_id', $this->cikl\_id)*

*->where('subject\_id', $this->subject\_id)*

*->where('subgroup', '<>', 2)*

*->sum('practice') + static::where('group\_id', $this->group\_id)*

*->where('subject\_id', $this->subject\_id)*

*->where('subgroup', '<>', 2)*

*->sum('lab');*

*}*

*public function getTheoryMainAttribute()*

*{*

*return static::where('group\_id', $this->group\_id)*

*->where('cikl\_id', $this->cikl\_id)*

*->where('subject\_id', $this->subject\_id)*

*->where('subgroup', '<>', 2)*

*->sum('theory');*

*}*

*public function getTotalMainAttribute()*

*{*

*return static::where('group\_id', $this->group\_id)*

*->where('cikl\_id', $this->cikl\_id)*

*->where('subject\_id', $this->subject\_id)*

*->where('subgroup', '<>', 2)*

*->sum('total');*

*}*

*public function getMainAttribute()*

*{*

*return static::where('group\_id', $this->group\_id)*

*->where('cikl\_id', $this->cikl\_id)*

*->where('subject\_id', $this->subject\_id)*

*->where('year', $this->year)*

*->where('semestr', $this->semestr)*

*->where('subgroup', '<>', 2)*

*->first();*

*}*

*public function results()*

*{*

*return $this->hasMany('App\Result');*

*}*

*public function generateLessons()*

*{*

*$left = $this->total;*

*$order = 0;*

*foreach($this->lessons as $lesson) {*

*if(!$left) {*

*$lesson->delete();*

*continue;*

*}*

*if($this->subject->divide == 1 || $this->subgroup == 2) {*

*$lesson->subgroup = $this->subgroup;*

*}*

*$lesson->group\_id = $this->group\_id;*

*$lesson->save();*

*$left -= $lesson->total;*

*$order ++;*

*}*

*while ($left > 0) {*

*$given = $left < 2 ? 1 : 2;*

*$lesson = Lesson::create([*

*'plan\_id' => $this->id,*

*'group\_id' => $this->group\_id,*

*'total' => $given,*

*'order' => ++$order,*

*]);*

*if($this->subject->divide == 1 || $this->subgroup == 2) {*

*$lesson->subgroup = $this->subgroup;*

*}*

*$lesson->group\_id = $this->group\_id;*

*$lesson->save();*

*$left -= $given;*

*}*

*return true;*

*}*

*public function generateRatings()*

*{*

*foreach($this->group->students as $s) {*

*foreach($this->lessons as $l) {*

*if(!$l->subgroup || $l->subgroup == $s->subgroup) {*

*$rating = Rating::updateOrCreate([*

*'student\_id' => $s->id,*

*'lesson\_id' => $l->id,*

*]);*

*$rating->save();*

*} else {*

*Rating::where([*

*'student\_id' => $s->id,*

*'lesson\_id' => $l->id,*

*])->delete();*

*}*

*}*

*}*

*}*

*public function generateResults()*

*{*

*if(in\_array($this->cikl\_id, [7,8,9])) return;*

*$plan = $this;*

*if($this->subject->divide == 2 && $this->main) {*

*$plan = $this->main;*

*$plan->subgroup = 0;*

*}*

*foreach ($this->group->students as $key => $student) {*

*if($plan->subgroup == $student->subgroup || !$plan->subgroup) {*

*$result = Result::updateOrCreate([*

*'plan\_id' => $plan->id,*

*'student\_id' => $student->id*

*]);*

*$result->save();*

*} else {*

*Result::where([*

*'plan\_id' => $plan->id,*

*'student\_id' => $student->id*

*])->delete();*

*}*

*}*

*}*

*public function checkNext($date)*

*{*

*$next = $this->lessons()->whereNull('date')->orWhere('date', $date)->orderBy('order', 'asc')->exists();*

*if(!$next) return false;*

*if($this->subgroup == 2 && $this->subject->divide == 2 && !empty($this->main)) {*

*$nextMain = $this->main->lessons()->where('date', $date)->orderBy('order', 'asc')->first();*

*if(!$nextMain) {*

*$nextMain = $this->main->lessons()->whereNull('date')->orderBy('order', 'asc')->first();*

*}*

*return $nextMain && $nextMain->practice > 0;*

*}*

*return true;*

*}*

*}*

Файл app/Http/Controllers/ResultController.php

*<?php*

*namespace App\Http\Controllers;*

*use App\Result;*

*use App\Teacher;*

*use App\Group;*

*use App\Student;*

*use App\Plan;*

*use Illuminate\Http\Request;*

*class ResultController extends Controller*

*{*

*public function index()*

*{*

*$group = \Request::get('group');*

*$sem = \Request::get('sem');*

*$user = \Auth::user();*

*$dis = Plan::where(function($q) {*

*$q->whereHas('subject', function($sq) {*

*$sq->where('divide', '<>', 2);*

*})*

*->orWhere('subgroup', '<>', 2);*

*})*

*->whereNotIn('cikl\_id', [7,8,9])*

*->when($group, function($query, $group) {*

*$query->where('group\_id', $group);*

*})*

*->when($sem, function($query, $sem) {*

*$query->where('semestr', $sem);*

*})*

*->when($user->role == 'teacher', function($query) use($user) {*

*$query->where('teacher\_id', $user->person\_id);*

*})*

*->orderBy('year', 'asc')*

*->orderBy('semestr', 'asc')*

*->orderBy('subject\_id', 'asc')*

*->orderBy('subgroup', 'asc')*

*->paginate(30);*

*return view('result.index', [*

*'dis' => $dis,*

*'groups' => Group::orderBy('name')->get(),*

*]);*

*}*

*public function edit($id)*

*{*

*$plan = Plan::findOrFail($id);*

*$plan->generateResults();*

*$results = Result::where('plan\_id', $id)->get();*

*return view('result.edit', [*

*'plan' => $plan,*

*'results' => $results,*

*]);*

*}*

*public function store(Request $request)*

*{*

*foreach ($request->results as $id => $res) {*

*$result = Result::find($id);*

*$result->fill($res);*

*$result->save();*

*}*

*}*

*public function view($id)*

*{*

*$student = Student::findOrFail($id);*

*$plans = Plan::where('group\_id', $student->group\_id)*

*->whereIn('subgroup', [0, $student->subgroup])*

*->whereNotIn('cikl\_id', [7,8,9])*

*->where('semestr', '<=', $student->group->kurs \* 2)*

*->orderBy('semestr', 'asc')->get();*

*$zachetka = [];*

*foreach ($plans as $key => $plan) {*

*$result = Result::updateOrCreate([*

*'plan\_id' => $plan->id,*

*'student\_id' => $id*

*]);*

*$zachetka[$plan->semestr][] = $result;*

*}*

*ksort($zachetka);*

*return view('result.view', [*

*'student' => $student,*

*'zachetka' => $zachetka,*

*]);*

*}*

*}*

Файл resources/views/journal/view.blade.php

*@extends('layouts.app')*

*@section('title', 'Журнал | '.$plan->subject->name)*

*@section('content')*

*<div class="alert alert-success col-sm-6" id="saved" hidden style="position: fixed;z-index: 999;top: 60px;right: 15px;">*

*Сохранено*

*</div>*

*<h3>{{ $plan->subject->name }}</h3>*

*<hr>*

*<div class="d-flex justify-content-between">*

*<div>*

*<p>Преподаватель: {{ $plan->teacher->fullName }}</p>*

*<p>Учебный год: {{ ($plan->year).'-'.($plan->year + 1) }}</p>*

*<p>Семестр: {{ $plan->semestr % 2 ? 1 : 2 }}</p>*

*<p>{{ $plan->kurs }} курс, группа {{ $plan->group->codes[$plan->kurs] }}</p>*

*</div>*

*<div>*

*<a href="/journal/{{ $plan->id }}/refresh" class="btn btn-sm btn-outline-secondary self-reload">Обновить</a>*

*<button id="save" class="btn btn-sm btn-outline-primary">Сохранить</button>*

*</div>*

*</div>*

*<div class="table-responsive">*

*<table class="table table-bordered table-sm">*

*<thead>*

*<tr>*

*<th>№</th>*

*<th>ФИО студента</th>*

*@foreach($plan->lessons as $l)*

*<th style="min-width: 32px"><small>{{ $l->date ? date('d.m', strtotime($l->date)) : '' }}</small></th>*

*@endforeach*

*</tr>*

*</thead>*

*<tbody>*

*@foreach($students as $key => $s)*

*<tr>*

*<td>{{ $key + 1 }}</td>*

*<td class="text-nowrap">{{ $s->shortName }}</td>*

*@foreach($plan->lessons as $l)*

*<td {{ isset($l->list[$s->id]) ? 'data-id='.$l->list[$s->id]->id : '' }}*

*{{ isset($l->list[$s->id]) && $l->date ? 'contenteditable=true' : '' }}*

*class="text-center {{ isset($l->list[$s->id]) ? '' : 'table-active' }}">*

*{{ @$l->list[$s->id]->value }}{{ @$l->list[$s->id]->miss ? 'н' : '' }}*

*</td>*

*@endforeach*

*</tr>*

*@endforeach*

*</tbody>*

*</table>*

*</div>*

*@endsection*

*@section('scripts')*

*<script type="text/javascript">*

*document.querySelectorAll('[contenteditable]').forEach(el => el.onkeydown = function(event) {*

*if([37,38,39,40].includes(event.keyCode)) {*

*event.preventDefault()*

*var tr, td*

*switch(event.keyCode) {*

*case 40:*

*tr = event.path[1].nextElementSibling*

*if(tr) {*

*td = tr.cells[event.path[0].cellIndex]*

*if(td) td.focus()*

*}*

*break;*

*case 38:*

*tr = event.path[1].previousElementSibling*

*if(tr) {*

*td = tr.cells[event.path[0].cellIndex]*

*if(td) td.focus()*

*}*

*break;*

*case 39:*

*tr = event.path[1]*

*if(tr) {*

*td = tr.cells[event.path[0].cellIndex + 1]*

*if(td) td.focus()*

*}*

*break;*

*case 37:*

*tr = event.path[1]*

*if(tr) {*

*td = tr.cells[event.path[0].cellIndex - 1]*

*if(td) td.focus()*

*}*

*break;*

*}*

*}*

*})*

*document.getElementById('save').onclick = function() {*

*var table = document.querySelectorAll('td[data-id]')*

*var data = []*

*for(var i = 0; i< table.length; i++) {*

*data.push({*

*id: table[i].dataset.id,*

*value: table[i].innerHTML*

*})*

*}*

*axios.post('/journal', data)*

*.then(response => {*

*document.getElementById('saved').hidden = false*

*setTimeout(() => document.getElementById('saved').hidden = true, 3000)*

*})*

*}*

*</script>*

*@endsection*

**ПРИЛОЖЕНИЕ Б**

**СОДЕРЖАНИЕ ДИСКА**

Папка «project» – совокупность исходных файлов по разработке проекта. Объем папки занимает 59,2 Mb.

Файл «Афанасьева Дина.docx» – пояснительная записка к дипломному проекту в текстовом формате «\*.docx». Занимаемый объем – 2,22 Mb.