МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет вычислительной техники

Кафедра «Вычислительная техника»

Специальность: 09.05.01 «Применение и эксплуатация автоматизированных систем специального назначения»

Специализация: «Эксплуатация вычислительных машин, комплексов, систем и сетей специального назначения»

ДИПЛОМНЫЙ ПРОЕКТ

на тему

Планировщик задач для инклюзивного образования

Студент 03.06.2020 (подпись, дата) Кузнецов В.А. (ФИО полностью)

Руководитель 03.06.2020 (подпись, дата) Дубинин В.Н. (фамилия, инициалы)

Нормоконтролёр 03.06.2020 (подпись, дата) Кучин А.В. (фамилия, инициалы)

Рецензент (должность, место работы) 03.06.2020 (подпись, дата) Севастьянов А.В. (фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 10.06.2020 г. № 13)

Заведующий кафедрой (подпись) М.А. Митрохин (фамилия, инициалы)

Работа защищена с отметкой отлично (протокол заседания ГЭК от 18.06.2020 № 7)

Секретарь ГЭК Е.И. Гурин

Пенза 2020

Федеральное государственное бюджетное образовательное учреждение
высшего образования
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет	*Кафедра*
Вычислительной техники	Вычислительная техника

«Утверждаю»
Заведующий кафедрой
_____М.А. Митрохин
«30» __апреля__ 2020 г.

ЗАДАНИЕ
НА ДИПЛОМНЫЙ ПРОЕКТ

1. Кузнецов Вадим Александрович гр. 15ВО2 факультета вычислительной техники.

2.Тема работы: «Планировщик задач для инклюзивного образования».

Тема утверждена приказом ПГУ № 0335а/о от 30.04.2020

3. Руководитель работы к.т.н., доцент каф. ВТ ПГУ Дубинин В.Н.

4. Задание на работу (назначение разработки, исходные данные и т.п.).

Система предназначена для планирования задач для инклюзивного образования.

5. Перечень подлежащих разработке вопросов.

Требуется произвести анализ предметной области, выбрать функциональный состав системы, разработать модели системы, базу данных и создать шаблоны, разработать автоматизированную систему планирования задач. Система должна осуществлять создание расписания, мероприятий, их изменение и удаление, а также их просмотр от разных пользователей. В технологической части необходимо представить программное обеспечение, в котором разрабатывалась система планирования задач. Также выполнить задание по БЖД и экономике.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Анализ предметной области	85	10.02.2020-20.02.2020	
2.	Разработка моделей и базы данных, а также шаблонов	95	21.02.2020-03.03.2020	
3.	Разработка web-приложения	180	04.03.2020-25.03.2020	
4.	Расчет экономической части	95	26.03.2020-05.04.2020	
5.	Исправление ошибок в работе web-приложения	204	06.04.2020-06.05.2020	
6.	Оформление дипломной работы	205	07.05.2020-02.06.2020	

Дата выдачи задания "29" апреля 2020 г.

Руководитель работы 30.04.2020 Дубинин В.Н.
(подпись, дата)

Задание к исполнению принял студент Кузнецов В.А.
(подпись, дата)

Работу к защите допустить
Декан факультета Фионова Л.Р.

РЕФЕРАТ

Пояснительная записка содержит 144 листов, 38 рисунков, 15 таблиц, 20 источников, 3 приложения.

ПЛАНИРОВЩИК ЗАДАЧ ДЛЯ ИНКЛЮЗИВНОГО ОБРАЗОВАНИЯ, БАЗА ДАННЫХ, REACT, МОДЕЛЬ, UML-ДИАГРАММА, WEB-ПРИЛОЖЕНИЕ.

Объектом исследования является планировщик задач.

Цель работы – разработка программного обеспечения для инклюзивного образования.

Разработка велась на языке PHP.

В процессе разработки был проведен анализ требований к разрабатываемому продукту, анализ существующих современных средств проектирования программных продуктов, анализ аналогичных систем, изучены способы реализации WEB проектов.

В результате выполнения дипломного проекта было разработано программное обеспечение, осуществляющее составление плана мероприятий и расписания занятий. Осуществлено функциональное тестирование разработанного приложения, которое показало корректность его работы.

В проектной части описаны выполненные функции приложения.

В технологической части описана технология разработки и отладки приложения.

В экономической части дается технико-экономическое обоснование целесообразности разработки системы.

В разделе безопасности жизнедеятельности описаны психофизиологические факторы, воздействующие на оператора ЭВМ, а так же рассмотрено воздействие работы с ПК на зрение человека. Графическая часть содержит схемы всей системы, данных, структуры базы данных, алгоритма работы программных модулей.

Содержание

РЕФЕРАТ 2

Введение 6

1 Предпроектный анализ 9

1.2. Анализ существующих автоматизированных систем дляпланирования задач 10

«Битрикс24» 10

«Timetable» 11

Виртуальная среда МБОУ СОШ №20 12

«Remember The Milk» 13

2 Разработка автоматизированной системы для планирования задач в информационной системе для инклюзивного образования 14

2.1 Формализация задачи 14

2.2 Архитектура программно-аппаратного комплекса 20

2.3 Разработка структуры данных 21

2.4 Разработка программной части 25

2.4 Разработка модели 32

2.5 Формирование шаблонов страниц пользовательского интерфейса на основе модели GOMS 33

2.6 Описание работы приложения 40

3 Тестирование и развертка программного обеспечения 44

3.1 Создание проекта в NetBeans 45

3.2 Отладка кода с помощью XDebug 49

3.2.1 Как работает отладка 50

4 Технико-экономическое обоснование разработки программного обеспечения дипломного проекта 53
4.1 Организация создания программного обеспечения 53
4.2 Расчет цены программного обеспечения 53
4.2.1 Определение затрат на разработку программного обеспечения 54
4.2.2 Установление цены программного обеспечения 57
4.2.3 Установление нижнего предела цены с учетом тиражирования 58
5 Безопасность жизнедеятельности 59
Заключение 76
Список использованных источников 77
КОД ПРОГРАММЫ (СЕРВЕРНАЯ ЧАСТЬ) 79
КОД ПРОГРАММЫ (КЛИЕНТСКАЯ ЧАСТЬ) 91
ПРЕЗЕНТАЦИЯ 140

Введение

В настоящее время большую роль играет повсеместное распространение интернета. Благодаря этому абсолютно любой человек может, не выходя из дома, работать с большими массивами информации, а также осуществлять взаимодействие с другими людьми: как для дистанционной работы, так и для общения. Распространение интернета позволило получить доступ в сеть почти в каждой стране, поэтому люди во многих городах получили возможность иначе работать с информацией. Одновременно с распространением интернета у человека появляются все более и более современные многофункциональные и высокопроизводительные технические устройства: от настольных персональных компьютеров и заканчивая смарт-часами.

На особую роль можно выделить такие устройства, как смартфоны, планшеты и ноутбуки. Они выделяются из общей массы технических средств, потому что предоставляют функцию входа в сеть буквально «на ходу». Ведь в современном мире есть такой важный фактор, как мобильность. Но в школах до сих пор работают с бумажными носителями. В такую категорию входят: приказы, журналы успеваемости, дневники, приказы и прочие документы. Они не обеспечивают нужную мобильность, о которой говорится ранее. А так же у бумажных носителей есть ряд проблем и недостатков:

- при большом количестве документов, накопившихся со временем, имеют большой вес;
- занимают большое пронстранство;
- для них требуется особые условия хранения.

Чтобы упростить работу и обеспечить необходимую мобильность рекомендуется переходить на электронные носители информации, потому что решается много проблем, таких как:

- не занимается много места;
- легко копируются данные, тем самым решается проблема потери информации;
- позволяют дистанционно получить доступ к данным;
- может хранить в себе большое количество информации.

Актуальность данной работы обусловлена необходимостью перехода школ с бумажных носителей на электронные, используя системы, упрощающие операции по котролю успеваемости, документооборотов и другими функциями.

Объектом исследования является планировщик задач для инклюзивного образования.

Целью работы является разработка планировщика задач, отвечающей требованиям, необходимых для выполнения составления планов мероприятий и учебного процесса для классов и учителей, а также позволяющей хранить большой объем информации и получать доступ к ним по средствам сети интернет.

В связи с большим объемом информации, целесообразно использовать базы данных для хранения списка учеников и учителей, классов, расписания занятий и планирования мероприятий.

Большое количество расчетных операций делают целесообразным использование компьютеров для организации автоматизированной системы планирования задач для инклюзивного образования с применением базы данных.

Использование автоматизированных систем позволит родителям, ученикам и учителям проще воспринимать информацию о занятиях и ближайших мероприятиях, которые организует школа.

Темой дипломного проекта является разработка системы обработки информации для расписания и мероприятий. Эта система ориентируется на упрощение воспринимаемой информации.

Система должна использовать современную СУБД для обеспечения хранения всего спектра данных и система должна иметь возможность размещения на информационных серверах в компьютерных сетях различной конфигурации и для ее работы использовать унифицированные средства взаимодействия вычислительной системы пользователя с удаленным сервером.

Эффект от использования подобного средства будет достигнут за счет сокращения времени осуществлении выписки всей необходимой информации школы.

1 Предпроектный анализ

Тема дипломного проекта – планировщик задач для инклюзивного образования. Программное обеспечение представляет автоматизированную информационную систему учителей и учеников для получения массивов данных о мероприятиях и расписании.

Разработка программного комплекса ведется на основании задания на дипломное проектирование, и утверждено кафедрой "Вычислительная техника" ФГБОУ ВО ПГУ.

Решение задач для создания системы планировщика связано с участием учеников, учителей и другой категории людей и характеризуется значительной трудоемкостью восприятия информации.

При разработке программного обеспечения для решения перечисленных задач следует учесть:

- необходимость работы с расписанием занятий;
- возможность редактирования расписания занятий;
- обеспечение удобного интерфейса;
- необходимость контроля действий пользователя;
- возможность получения информации о мероприятиях и новостях школы;
- возможность работы с современными телекоммуникационными технологиями.

Разрабатываемая система предназначена для категории разных людей, чтобы они могли просто воспринимать информацию благодаря понятному интерфейсу:

- ведение новостной ленты мероприятий и новостей;
- регулирование расписания;

- сокращение времени для подачи информации ученикам, учителям и другим людям.

Система должна функционировать в кросплатформенном операционном пространстве. Для работы с программой обслуживающему персоналу достаточно знаний персонального компьютера на уровне умения работы с операционной системой (ОС) Windows 8 и приложениями написанных под данную ОС.

1.2. Анализ существующих автоматизированных систем

В настоящее время есть много веб-страниц и приложений для планирования задач или же составления расписания для школьных учереждений. Также есть сайты школ, где предоставлена информация о школе и электронный дневник учеников, для предаставлении информации как учеников, так и родителей о успеваемости.

Однако практика показала, что один и тот же программный продукт не может соответствовать полностью требованиям разных организаций и требует не только общей но и «тонкой» настройки.

Существует множество веб-страниц, приложений для компьютеров, а также и для телефонов, но нет таких, которые были специализированные для учебных учереждений, где указывалось расписание, новостная лента и ближашие мероприятия.

Ниже приведен обзор известных на сегодняшний день планировщиков задач, приложений и веб-страниц школ.

«Битрикс24»

Одним из основных фирм-разработчиков программного обеспечения для планирования задач является «1С-Битрикс».

Программное обеспечение «Битрикс24» было запущено 12 апреля 2012 года. Разработана в на базе php и доступен для пользователей в следующих операционных системах:

- macOS;
- Linux;
- IOS;
- Android;
- Windows;

А также можно можно получить доступ из любого браузера.

Спектр разработок этой фирмы обширен, но это не отменяет того, что она является заслуженным лидером в сфере менеджмента задач.

Основными функциональными особенностями «Битрикс24» являются:

- формирование чел-листов, где определяется важность задач и регулирование выполненных задач;
- регулирование задач, их группировка, сортировка;
- определение ролей для выполнения какой-либо задачи, где формируются права доступа;
- создание шаблонов проектов, отправка их подрядчикам;
- голосовые помощники, которые позволяют упростить работу с менеджентом задач.

Есть бесплатная версия, которая предоставляет минимальный спектр инструментов для работы программного обеспечения, но платная версия будет стоит за год 36 тысяч рублей без учета оплаты специалиста. Систему нельзя ставить в школе, потому что она предназначена для малого, среднего и большого бизнеса, избыток функций обеспечат лишние траты.

«Timetable»

Программа «Timetable» является мобильным приложением для простого составления расписания. Имеет удобный интерфейс и разработана компанией Gabriel Ittner.

Разработана программа была в 2012 году для студентов и учеников.

Такое приложение имеет некоторые преимущества:

- беззвучный режим телефона активируется автоматически, когда начинаются занятия;
- сортировка расписания и заданий;
- двухнедельный цикл расписания;
- уведомления о начале занятий.

Недостатком такого программного обеспечения является:

- невозможность синхронизироваться с календарем.гугл;
- имеет много ошибок в системном коде, поэтому работа приложения тормозит или выходит из него;
- уже давно не поддерживается разработчиком, поэтому ученикам приходится самим модернизировать приложение.

Приложение предназначена для индивидуальной работы и не сгодится для внедрения в школьную среду.

Виртуальная среда МБОУ СОШ №20

Виртуальная среда МБОУС СОШ №20 была сделана в 2007 году.

Веб-страница созданная для получения массивов данных:

- новостная лента;
- информация о школе, учебном плане;
- в личном кабинете можно увидеть электронный дневник, где будут указаны оценки;
- множество информации о безопасноти жизнидеятельности.

Интерфейс простой и позволяет любому пользователю найти нужную информацию, есть функция для слабовидящих, где убираются лишние элементы и все написано большими буквами.

К недостаткам веб-страницы МБОУ СОШ №20 можно причислить:

- устаревший дизайн;
- неправильное расставление элементов, от чего информация воспринимается тяжело, потому что слишком много данных представлено на одной лишь странице;
- нет возможности редактировать данные в настоящем времени;
- долгая прогрузка всей информации.

«Remember The Milk»

Программа «Remember The Milk» является мобильным приложением для простого составления задач. Является бесплатным для пользователей. Имеет удобный интерфейсм и разработана компанией Remember The Milk. Приложение является бесплатным и обладает такими преимуществами, как:

- уведомления отсылаются на почту, Twitter и мессенджеры;
- добавление промежуточных задач;
- синхронизация с Dropbox и «Google Диск»;
- имеет функцию «умный лист», где задачи сортируются по разной степени важности и есть добавление в категорию «отложеные».

Существенным недостатком приложения является ограниченностью возможностей. Нет синхронизации с аккаунтами, адаптация на русский язык является не точным и для получения полнометражной версии нужно получить подписку за 2800 рублей в год. Приложение работает только на телефоне и не подходит для внедрения в школы. [4]

2 Разработка автоматизированной системы для обеспечения логистики предприятия

2.1 Формализация задачи

В дипломном проекте разработанн планировщик задач для инклюзивного образования.

При разработке программного обеспечения необходимо учитывать процесс движения данных в учебном процессе. При его анализе необходимо изучить формы входных и выходных данных, влияющих на разработку подсистемы хранения данных.

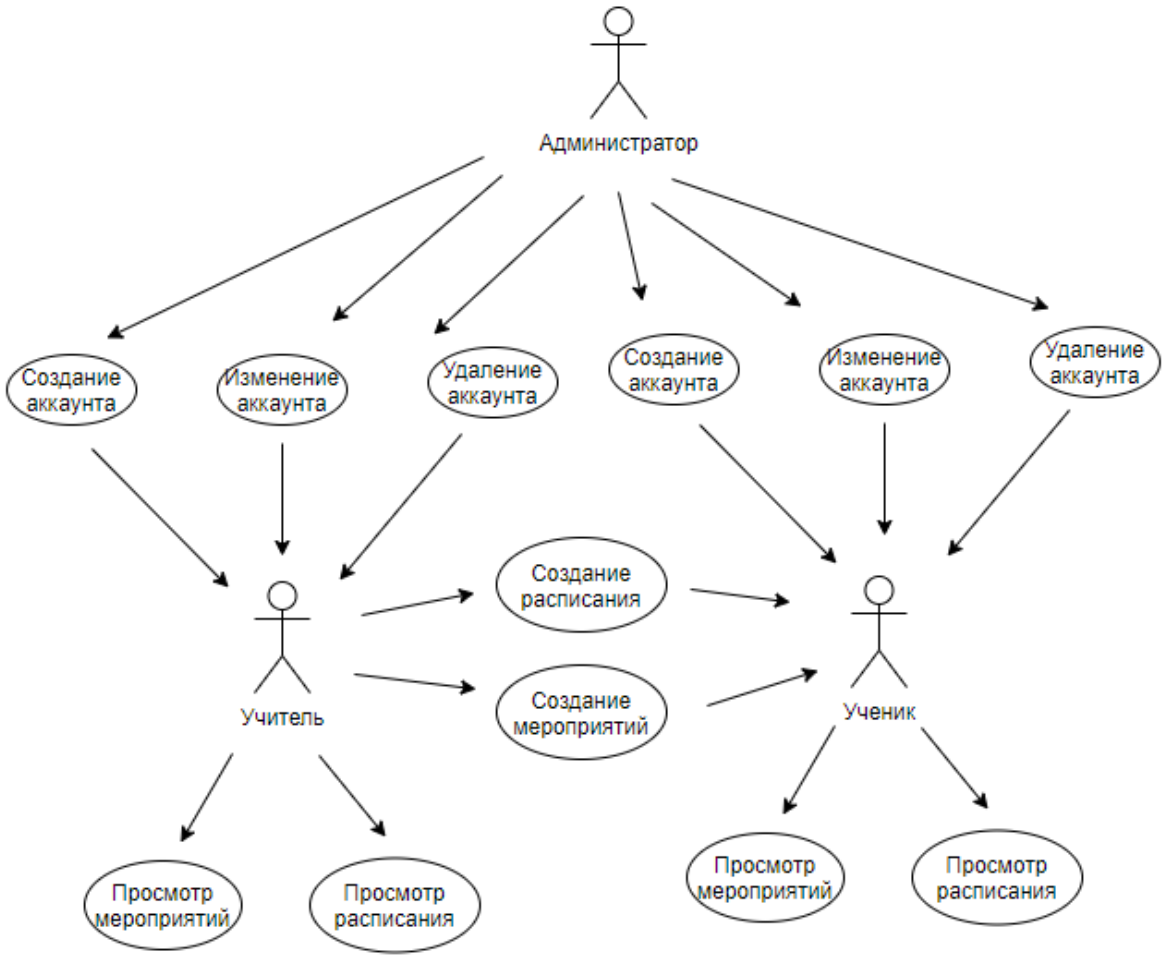


Рисунок 2.1 Схема движения данных и роли

Исходными или входными (первичными) документами для составления расписания и мероприятий служат:

- список расписания;
- список мероприятий;
- данные учеников;
- данные учителей;

Выходные документы должны формироваться по каждому учителю и ученику, а также мероприятиям и расписаниям.

Все программное обеспечение можно разделить на несколько частей: база данных, где хранятся все необходимые данные, программные модули для выполнения расчетов и визуальный интерфейс взаимодействия с пользователем. От правильной реализации каждого их компонентов программного комплекса зависит качество работы программного комплекса в целом. Основой построения такой программной системы является база данных. Для ее управления используют системы управления базами данных (СУБД). При проектировании используется СУБД MySQL.

СУБД — совокупность информационных массивов, технических, программных и языковых средств, предназначенных для сбора, хранения, поиска, обработки и выдачи данных по запросам конечных пользователей. Конечным называется пользователь, обращающийся к системе для получения необходимых ему данных. Необходимость использования СУБД обусловлена целью, упорядочить механизм хранения данных и исключить их потери.

В разрабатываемом программном обеспечении СУБД выполняет следующие функции:

- вход данных, используя формы ввода;
- формирование отчетных документов.

Проектирование базы данных является основной задачей при разработке программной системы. Основные компоненты базы данных являются

связующими частями между функциональными частями системы, так как входные данные одной из частей системы являются входными данными для другой. Проектирование охватывает три основные области:

- Проектирование конкретных объектов, которые будут реализованы в базе данных. Такими объектами являются таблицы, представления, индексы и хранимые процедуры.
- Проектирование конкретных экранных форм, отчетов и программ, которые будут сопровождать данные в базе данных, и обеспечивать выполнение запросов к этим данным.
- При определенных обстоятельствах в процессе проектирования также необходимо учитывать конкретную среду или технологию – например, топологию сети, конфигурацию аппаратных средств, использование реляционной СУБД, параллельной обработки и распределенной архитектуры базы данных.

При проектировании базы данных одной из важнейших является задача выделения различных уровней представления и описания данных. Известно, что одна и та же информация может быть представлена различными способами. Существуют несколько уровней представления данных:

1. Внешнее представление – это совокупность информационных моделей, каждая из которых описывает ту часть предметной области, которая необходима для обеспечения информационных потребителей определенного приложения;
2. Концептуальное представление – обобщение и согласование внешних представлений пользователей, т.е. оно отображает общие информационные потребности всех приложений, согласованных между собой. Концептуальное представление системно независимо, т.е. независимо от конкретной СУБД, операционной системы и аппаратного обеспечения ЭВМ.

3. Представление в терминах конкретной СУБД. Все современные типы ЭВМ поддерживают определенный тип модели данных, в основном сетевой, иерархический или реляционный. Описание предметной области в терминах модели данных, поддерживаемой конкретной СУБД, называется логическим представлением.

Внутреннее представление – это представление о размещении данных в среде хранения и путях доступа к ним. При этом должны быть определены различные типы хранимых записей, существующие индексы, представление хранимых полей, физическая последовательность хранимых записей и т.д.

При построении концептуальных моделей предметной области, используя большинство предлагаемых методик, на этапе построения концептуальной модели выполняются следующие задачи:

1. Выявляются цели, которые преследует администратор предприятия, принимая решения о создании информационной системы.
2. Выделяется на основе этих целей потенциальных пользователей системы.
3. Создаются внешние пользовательские представления.
4. Выявляются и устраняются конфликтные ситуации, возникающие между различными внешними представлениями, и интегрирует их в концептуальное представление.
5. Записывается концептуальное представление на формальном языке, позволяющем легко отображать это представление в одну из логических моделей данных.

Формирование концептуальной модели базы данных включает в себя:

- Идентификацию функциональной деятельности заданной предметной области;
- Идентификацию объектов, которые осуществляют эту функциональную деятельность, и формирование из этих операций

последовательность событий, которые могут идентифицировать все сущности и взаимосвязи между ними;

- Идентификацию взаимосвязей между сущностям;

Для того чтобы спроектировать структуру базы данных, необходима исходная информация о предметной области. Желательно, чтобы эта информация была представлена в формализованном виде. Для этих целей используют искусственные формализованные языковые средства.

Под инфологической моделью понимают описание предметной области, выполненное с использованием специальных языковых средств, не зависящих от используемых в дальнейшем программных средств. Для её описания используются как языки аналитического (описательного) типа, так и графические средства. Последние в настоящее время приобретают все большую популярность. Существует ряд средств автоматизации проектирования, для которых исходные данные представляются в виде совокупности графических схем. Графическое представление является наиболее простым для восприятия и анализа, поэтому воспользуемся графическим способом отображения модели "объект - свойство - отношение". В дипломном проекте используется модель «сущность-связь» (ER - модель) как наиболее распространенная, обеспечивающая вполне достаточное и адекватное описание объектов реального мира, их свойств и взаимосвязей. В приложении Б приведена схема разработанной базы данных.

Визуальная система пользователя предоставляет возможность взаимодействовать с СУБД посредством графического интерфейса. В СУБД MySQL графический пользовательский интерфейс реализуется с помощью форм специальных клиентских плагинов (HeidiSQL).

Формы являются типом объектов базы данных, которые обычно используется для отображения данных в базе данных. Форму можно также использовать как кнопочную форму, открывающую другие формы или отчеты базы данных, а также как пользовательское диалоговое окно для ввода данных и выполнения действий, определяемых введенными данными.

Большинство форм являются присоединенными к одной или нескольким таблицам и из базы данных. Форма не должна включать все поля из каждой таблицы или запроса, на основе которых она создается. Присоединенная форма получает данные из базового источника записей. Другие выводящиеся в форме сведения, такие как заголовок, дата и номера страниц, сохраняются в макете формы. Графические элементы, такие как линии и прямоугольники, сохраняются в макете формы.

Связь между формой и ее источником записей создается при помощи графических объектов, которые называют. Наиболее часто используемым для вывода и ввода, данных типом элементов управления является поле.

Формы можно также открывать в режиме сводной таблицы или в режиме диаграммы для анализа данных. В этих режимах пользователи могут динамически изменять макет формы для изменения способа представления данных. Существует возможность упорядочивать заголовки строк и столбцов, а также применять фильтры к полям. При каждом изменении макета сводная форма немедленно выполняет вычисления заново в соответствии с новым расположением данных. В режиме сводной таблицы имеется возможность просматривать исходные данные или сводные данные, упорядочивать поля в областях фильтра, строк, столбцов и данных. Кроме этого можно визуально представлять данные, выбрав тип диаграммы и упорядочивая поля в областях фильтра, рядов, категорий и данных.

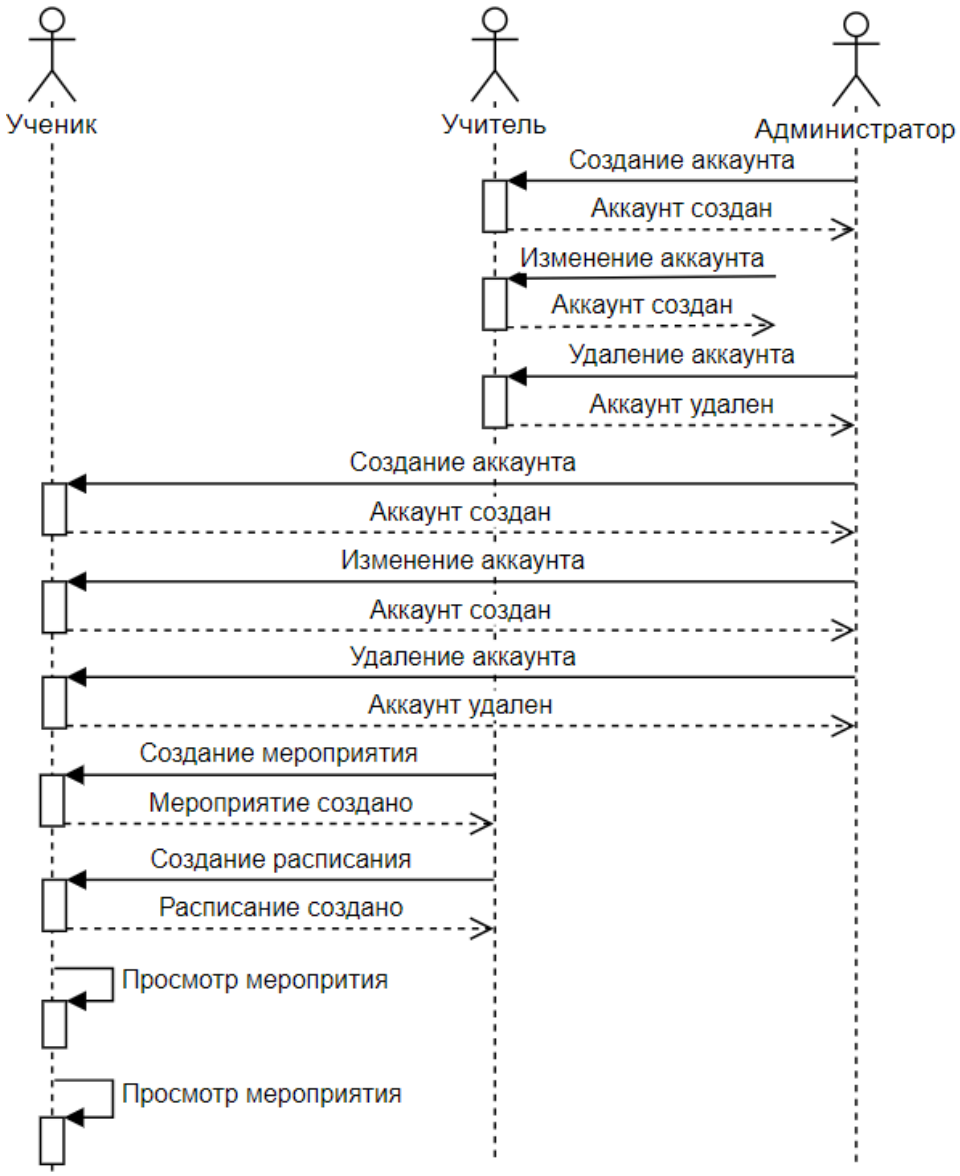


Рисунок 2.2 Схема движения данных и ролей.

2.2 Архитектура программно-аппаратного комплекса

Данный программный комплекс предназначен для развертывания на базе платформы Intel. Рекомендованная конфигурация для сетевой работы AT Pentium IV – 2 ГГц, с объемом оперативной памяти не менее 2 ГБ (рекомендуется 4 ГБ), постоянного запоминающего устройства не менее 500 ГБ, видеокарта с объемом памяти не менее 2 ГБ, дисковод CD-ROM, USB накопитель. Такие системные требования обусловлены применением требованиями ОС.

Для однопользовательской работы рекомендуется следующая конфигурация – IBM-совместимый компьютере типа IBM PC / AT Pentium IV Celeron – 1,7 ГГц, с объемом оперативной памяти не менее 2 ГБ (рекомендуется 4 ГБ), постоянного запоминающего устройства не менее 10 ГБ, видеокарта с объемом памяти не менее 128 Мб, дисковод CD-ROM, USB накопитель, монитор 17’’, средства мультимедиа (звуковая карта, колонки) для распечатки отчетной информации формата А4 рекомендуется любой тип принтера. Для работы пользователя необходима ОС Win32 - Windows 10 и наличие СУБД MySQL.

Сетевая версия программного комплекса может быть установлена на WEB серверах с ОС Windows или ОС UNIX, использоваться на бездисковых рабочих местах, оснащенных средствами мультимедиа.

2.3 Разработка структуры данных

На основе анализа предметной области поставленной задачи разработана структура базы данных в среде PostgresSQL.

База данных по учету движения товарно-материальных ценностей включает следующие таблицы:

Таблица 2.1 - priveleges – предназначена для обеспечения доступа учеников к расписанию и мероприятиям.

Наименование поля	Тип данных	Ключ	Назначение
Id_privelege	int	PK	Уникальный идентификатор привилегии
Id_schedules_types	int	FK	Уникальный идентификатор мероприятия

Продолжение таблицы 2.1

Наименование поля	Тип данных	Ключ	Назначение
Id_shedules	Int	FK	Уникальный идентификатор занятия
Id_user	int	PK,FK	Идентификатор пользователя

Таблица 2.2 - users – предназначена для хранения информации о пользователях.

Наименование поля	Тип данных	Ключ	Назначение
Id_user	Int	PK	Уникальный идентификатор пользователя
login	Varchar(25)	_	Логин учителя
password	Varchar(30)	_	Пароль учителя
F_Name	Varchar(25)	_	Имя учителя
L_Name	Varchar(20)	_	Фамилия учителя
S_Name	Varchar(20)	_	Отчество учителя

Таблица 2.3 - user_class – предназначена для хранения информации о учениках.

Наименование поля	Тип данных	Ключ	Назначение
Id_user_class	Int	PK	Уникальный идентификатор пользователя
loginClass	Varchar(25)	_	Логин ученика
passwordClass	Varchar(30)	_	Пароль ученика

Продолжение таблицы 2.3

Наименование поля	Тип данных	Ключ	Назначение
F_Student	Varchar(25)	_	Имя ученика
L_Student	Varchar(20)	_	Фамилия ученика
S_Student	Varchar(20)	_	Отчество ученика
Id_class	Int	PK,FK	Уникальный идентифатор класса

Таблица 2.4 - schedules – предназначена для хранения информации о расписании.

Наименование поля	Тип данных	Ключ	Назначение
Id_shedules	int	PK	Уникальный идентификатор занятия
Name	Varchar(255)	_	Наименование занятия
Time	Time()	_	Наименование время начала занятия

Таблица 2.5 - schedules_types – предназначена для хранения информации о мероприятиях.

Наименование поля	Тип данных	Ключ	Назначение
Id_schedules_types	int	PK	Уникальный идентификатор мероприятия

Продолжение таблицы 2.5

Наименование поля	Тип данных	Ключ	Назначение
Name_Types	Varchar(255)	_	Наименование мероприятия
Time_Types	Varchar(255)	_	Наименования время начала мероприятия

Таблица 2.6 - classes – предназначена для хранения информации о классах.

Наименование поля	Тип данных	Ключ	Назначение
Id_class	int	PK	Уникальный идентификатор класса
Name	Varchar(20)	_	Наименование класса

Таблица 2.7 - class_schedule – предназначена для хранения информации о доступе класса.

Наименование поля	Тип данных	Ключ	Назначение
Id_class_schedule	int	PK	Уникальный идентификатор категории
Id_class	int	PK,FK	Уникальный идентификатор класса
Id_schedules_types	int	FK	Уникальный идентификатор мероприятия
Id_shedules	int	FK	Уникальный идентификатор занятия

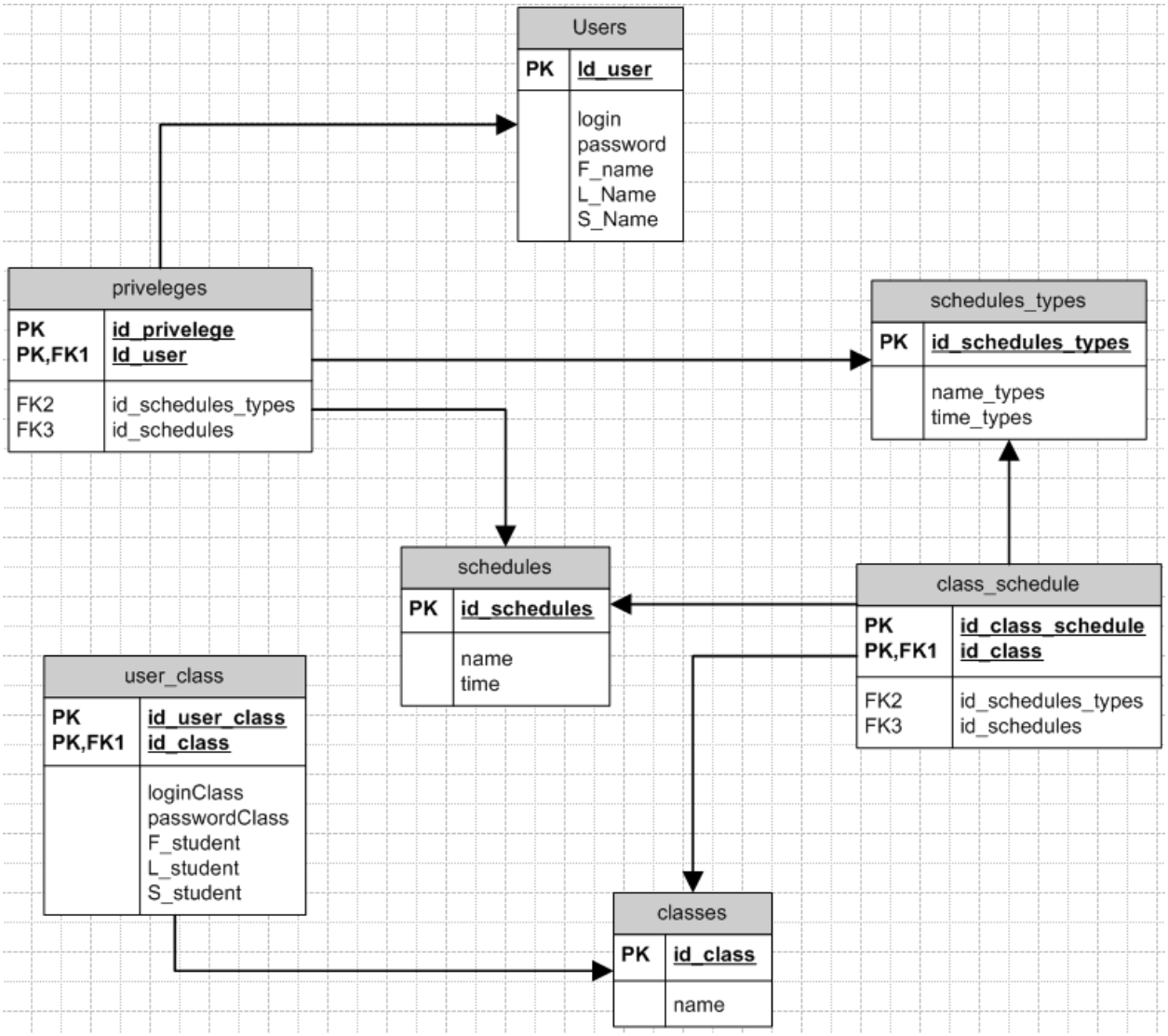


Рисунок 2.3 – Структура базы данных

2.4 Разработка программной части

Программное обеспечение разработано на основе паттерна MVC (Модель – Вид -Контроллер):

– Модель (Model) предоставляет данные и реагирует на команды контроллера, изменяя своё состояние;

– Представление (View) отвечает за отображение данных модели пользователю, реагируя на изменения модели;

– Контроллер (Controller) интерпретирует действия пользователя, оповещая модель о необходимости изменений.

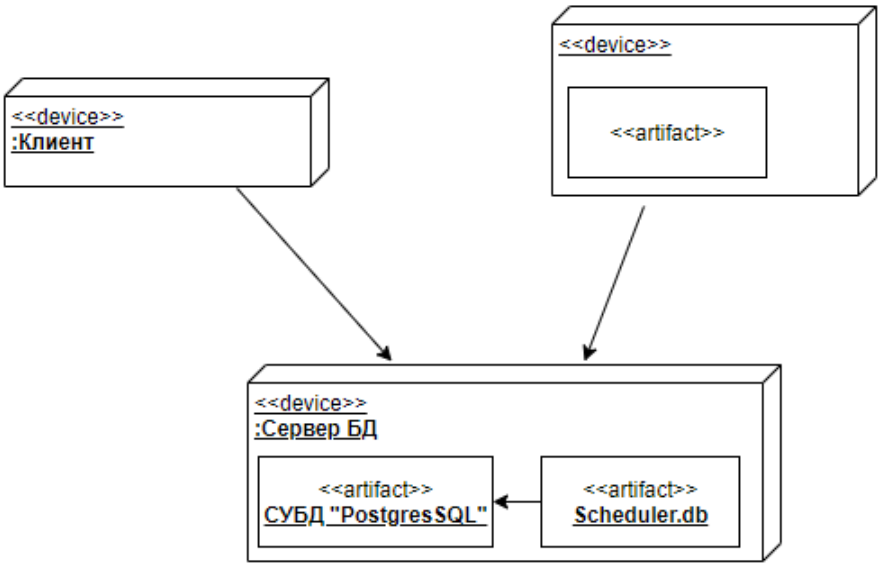


Рисунок 2.4 Сформированное приложение MVC

Фактически разработанное приложение состоит из следующих цепочек MVC – таблицы 2.8-2.11:

Таблица 2.8 - Авторизация

Модель	Представление	Контроллер
Отправка данных на сервер для авторизации. Отправка запроса на сервер для выхода.	Вывод экрана авторизации.	Ввод логина и пароля. Выход из пользователя.

Таблица 2.9 — Пользователи.

Модель	Представление	Контроллер
Отправка запроса на создание, изменение и удаление пользователя.	Вывод списка пользователей.	Изменение, создание и удаление пользователя.

Таблица 2.10 - Мероприятия

Модель	Представление	Контроллер
Отправка списка мероприятий на сервер.	Вывод списка мероприятий.	Добавление мероприятия и их изменение.

Таблица 2.11 — Расписание

Модель	Представление	Контроллер
Отправка списка занятий на сервер.	Вывод расписания.	Изменение расписания и его изменение.

2.5 Описание функций контроллера

Ниже описаны функции приложения.

Функция [set]Login реализует вход пользователя в систему и создаёт сессию, входными данными является логин пользователя и пароль, выходные данные отсутствуют.

Функция [get]Logout реализует выход пользователя из системы и завершение сессии, выходные и выходные данные отсутствуют.

Функция [get]User_classes реализует вывод данных о пользователях, входные данные отсутствуют, выходными данными являются имя, фамилия и отчество ученика.

Функция [get]User_class реализует вывод данных о текущем пользователе, входные данные отсутствуют, выходными данными являются: уникальный идентификатор пользователя, имя, фамилия, отчество ученика, уникальный идентификатор класса.

Функция [set]User_class реализует создание пользователя, входными данными являются: логин, пароль, имя, фамилия отчетство ученика и класс, выходные данные отсутствуют.

Функция [change]User_class реализует изменение данных пользователя, входными данными являются уникальный идентификатор пользователя, логин, пароль, имя, фамилия отчетство ученика и класс.

Функция [delete]User_class реализует удаление пользователя, входными данными является уникальный идентификатор пользователя, выходные параметры отсутствуют.

Функция [get]Users реализует вывод данных о пользователях, входные данные отсутствуют, выходными данными являются имя, фамилия и отчество учителя.

Функция [get]User реализует вывод данных о текущем пользователе, входные данные отсутствуют, выходными данными являются: уникальный идентификатор пользователя, логин, пароль, имя, фамилия, отчество учителя.

Функция [set]User реализует создание пользователя, входными данными являются: логин, пароль, имя, фамилия, отчетство учителя, выходные данные отсутствуют.

Функция [change]User реализует изменение данных пользователя, входными данными являются уникальный идентификатор пользователя, логин, пароль, имя, фамилия отчетство учителя.

Функция [delete]User реализует удаление пользователя, входными данными является уникальный идентификатор пользователя, выходные параметры отсутствуют.

Функция [set]Privilege реализует установку привилегии, входными параметрами является уникальный идентификатор пользователя, уникального индетификатора мероприятия и уникального индетификатора занятия, выходные данные отсутствуют.

Функция [delete]PrivilegeUser реализует удаление привилегии пользователя, входными параметрами является идентификационный номер пользователя, выходные параметры отсутствуют.

Функция [change]Privilege реализует изменение привилегий пользователя, входными параметрами являются идентификационный номер

пользователя, уникальный идентификатор номера мероприятия, уникальный идентификатор номера занятия, уникальный идентификатор номера привилегии.

Функция [get]PrivilegeByUser реализует вывод привилегий по пользователю входным параметром является идентификационный номер пользователя.

Функция [get]Privileges реализует вывод всех привилегий, входные и выходные параметры отсутствуют.

Функция [set]Schedules реализует добавление занятий в расписание, входными данными являются: Уникальный идентификатор занятия, наименования занятия, наименование время начала занятия, выходные данные отсутствуют.

Функция [change]Schedules реализует изменение данных расписания, входными данными являются: уникальный идентификатор занятия, наименование занятия, наименование время начала занятия, выходные данные отсутствуют.

Функция [delete]Schedules реализует удаление занятия из расписания, входными данными является уникальный идентификатор занятия, выходные данные отсутствуют.

Функция [get]Schedules_types реализует получение списка мероприятий, входные и выходные параметры отсутствуют.

Функция [set]Schedules_types реализует добавление мероприятий, входными данными являются: уникальный идентификатор мероприятия, наименование мероприятия, наименование время начала мероприятия, выходные данные отсутствуют.

Функция [change]Schedules_types реализует изменение данных о мероприятиях, входными данными являются: уникальный идентификатор, мероприятия, наименования мероприятия, выходные данные отсутствуют.

Функция [delete]Schedules_types реализует удаление мероприятия, входными данными является уникальный идентификатор мероприятия, выходные данные отсутствуют.

Функция [get]Classes реализует вывод всех классов, входные данные отсутствуют.

Функция [set]Classes реализует добавление классов, входными данными является уникальный идентификатор класса, выходные данные отсутствуют.

Функция [delete]Classes реализует удаление классов из списка, входными данными является уникальный идентификатор класса, выходные данные отсутствуют.

Функция [change]Brand реализует изменение бренда, входными параметрами является идентификационный номер, идентификационный номер производителя, наименование, выходные параметры отсутствуют.

Функция [get]class_schedule реализует вывод информации о доступе класса к мероприятиям и занятиям, входные данные отсутствуют.

Функция [set]class_schedule реализует добавление доступа для класса, входными данными являются: уникальный идентификатор категории, уникальный идентификатор класса, уникальный идентификатор мероприятия, уникальный идентификатор занятия, выходные данные отсутствуют.

Функция [change]class_schedule реализует изменение доступа для класса, входными данными являются: уникальный идентификатор категории, уникальный идентификатор класса, уникальный идентификатор мероприятия, уникальный идентификатор занятия, выходные данные отсутствуют.

Функция [delete]class_schedule реализует удаление доступа для класса, входными данными является уникальный идентификатор категории, выходные данные отсутствуют.

2.5 Разработка модели

Функции контроллера выполняют функции view, insert, delete, read, wrire для моделей.

Модели – это классы, реализованные на PHP, для связи вида и СУБД (рисунок 2.5).

Фактически данные классы (рисунок 2.5) транслируют данные в таблицы 2.1 -2.7.

В результате реализации приложения на основании паттерна Модель – Вид-Контроллер, реализуется приложение, структура которого показана на рисунке 2.5.

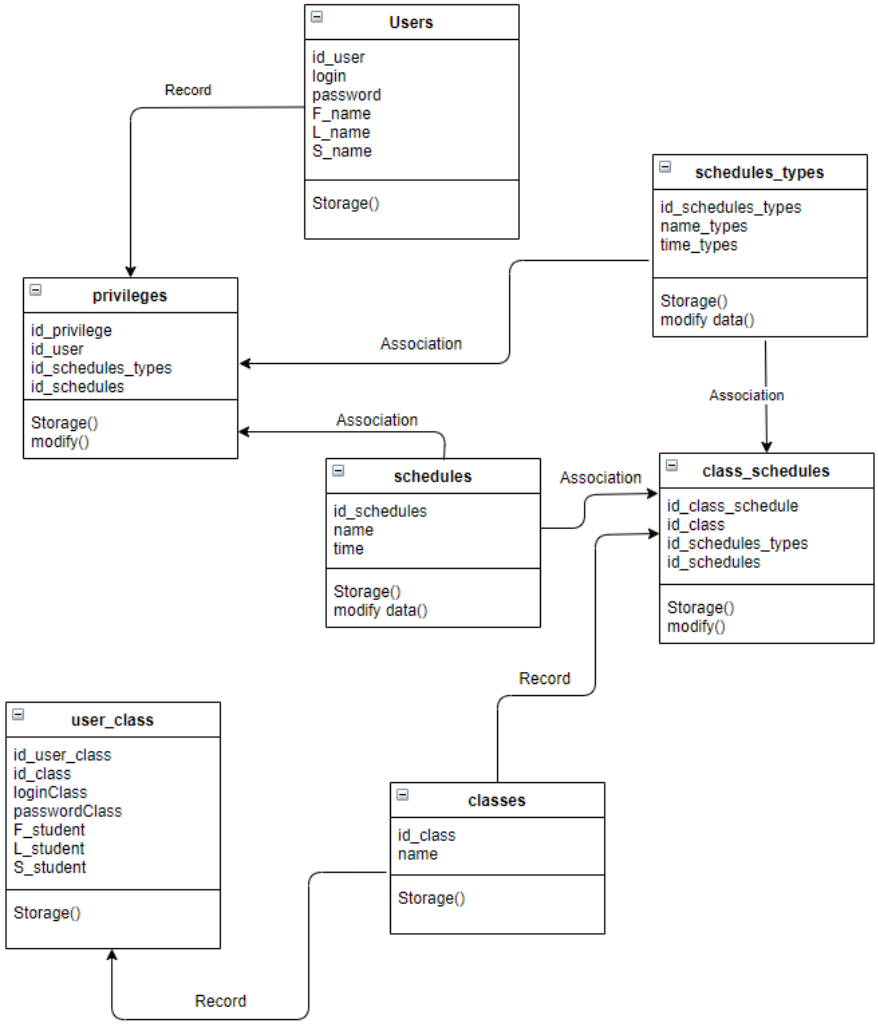


Рисунок 2.5 Классы моделей.

Данные и команды между блоками приложения передаются в формате JSON, с помощью стандартного механизма REST API.

Всё взаимодействие с сервером сводится к 4 операциям (4 - это необходимый и достаточный минимум, в конкретной реализации типов операций может быть больше):

Получение данных с сервера (в формате JSON): добавление новых данных на сервер, модификация существующих данных на сервере, удаление данных на сервере.

Операция получения данных не может приводить к изменению состояния сервера.

Для каждого типа операции используется свой метод HTTP-запроса: получение – GET, добавление – POST, модификация – PUT, удаление – DELETE.

2.6 Формирование шаблонов страниц пользовательского интерфейса на основе модели GOMS

Количественные методы помогают свести спорные вопросы в оценке качества интерфейса к простым вычислениям.

Одним из лучших подходов к количественному анализу моделей интерфейсов является классическая модель GOMS (the model of goals, objects, methods and selection rules) – модель целей, объектов, методов и выбор правил [9].

Эта модель основана на оценке скорости ввода. Время, требуемое для выполнения какой-то задачи системой пользователь-компьютер, является суммой всех временных интервалов, которые потребовались системе на выполнение элементарных действий, составляющих данную задачу.

В таблице 2.12 приведены стандартные средние интервалы для некоторых действий, выполняемых различными пользователями.

Для того чтобы определить, в какие моменты будут проходить ментальные операции, необходимо следовать некоторым правилам.

Правило 0. Начальная расстановка операторов М. Оператор М устанавливается перед всеми операторами К (нажатие клавиши) и Р, предназначенными для выбора команд, если Р указывает на аргументы этих команд оператор М не ставится.

Таблица 2.12 – Интервалы некоторых жестов, выполняемых пользователями персональных компьютеров

Название жеста	Условное обозначение	Среднее время, с
Нажатие клавиши	К	0,2
Указание (на какую-либо позицию на экране монитора)	Р	1,1
Перемещение руки с клавиатуры на мышь или с мыши на клавиатуру	Н	0,4
Ментальная задержка	М	1,35
Время задержки ответа компьютера	R	-

Правило 1. Удаление ожидаемых операторов М. Если оператор, следующий за оператором М ожидаемый с точки зрения оператора, предшествующего М, то этот оператор М может быть удален и последовательность РМК превращается в РК.

Правило 2. Удаление операторов М внутри когнитивных единиц.

Если строка типа «МКМКМК…» принадлежит когнитивной единице, то следует удалить все операторы М, кроме первого. Когнитивной единицей

является непрерывная последовательность вводимых символов, которые могут образовывать название команды или аргумент.

Правило 3. Удаление M перед последовательными разделителями. Если оператор K означает лишний разделитель, стоящий в конце когнитивной единицы (например разделитель команды, следующий сразу за разделителем аргумента этой команды) то следует удалить оператор M, стоящий перед ним.

Правило 4. Удаление операторов M, которые являются прерывателями команд. Если оператор K является разделителем, стоящим после постоянной строки (например, название команды или любая последовательность символов, которая каждый раз вводится в неизменном виде), то следует удалить оператор M, стоящий перед ним. Но если оператор K является разделителем для строки аргументов или любой другой изменяемой строки, то M следует сохранить перед ним.

Правило 5. Удаление перекрывающихся операторов M. Любую часть операторов M, которая перекрывает оператор R, означающий задержку, связанную с ожиданием ответа компьютера, учитывать не следует.

Если имеется модель интерфейса, то с помощью GOMS можно определить время, необходимое пользователю для выполнения любой, четко сформулированной задачи, для которой данный интерфейс предусмотрен. Однако модели анализа не могут дать ответ на вопрос о том, насколько быстро должен работать интерфейс. Чтобы ответить на него, необходимо воспользоваться мерой, применяемой в теории информации.

Чтобы сделать правильную оценку времени, необходимого для выполнения задачи с помощью самого быстрого интерфейса, прежде всего, следует определить минимально количество информации, которое пользователь должен ввести, чтобы выполнить задачу. Это минимальное количество не зависит от модели интерфейса. Если методы работы, используемые в предполагаемом интерфейсе, требуют введения такого количества информации, которое превышает минимальное, это означает, что пользователь делает лишнюю работу и поэтому интерфейс следует

усовершенствовать. С другой стороны, если от пользователя требуется ввести именно то количество информации, которое необходимо для выполнения задачи, то для этой задачи интерфейс нельзя сделать более производительным путем изменения количества информации. В этом случае пути улучшения интерфейса (а также много путей для ухудшения) все же остаются, но, по крайней мере, данная цель повышения производительности будет уже достигнута.

Информационно-теоретическая производительность определяется так же, как понятие производительности определяется в термодинамике – отношением мощности на выходе к мощности на входе процесса. Если в течение какого-то периода времени электрогенератор, работающий от двигателя мощностью в 1000 ватт, производить 820 ватт, то он имеет производительность 820/1000=0,82. Производительность также часто обозначается через проценты.

Информационная производительность интерфейса E определяется как отношение минимального количества информации, необходимого для выполнения задачи, к количеству информации, которое должен ввести пользователь. Так же как и в отношении физической производительности, параметр E может изменяться в пределах от 0 до 1. Если никакой работы для выполнения задачи не требуется или работа просто не производится, то производительность составляет 1 (чтобы избежать деления на 0).

Производительность E может равняться и 0 в случаях, когда пользователь должен ввести информацию, которая бесполезна для выполнения задачи.

В параметре E учитывается только информация, необходимая для задачи и информация, вводимая пользователем.

Так для выбора одного варианта из четырех возможных достаточно двух бит информации, из восьми – трех, из 16 – 4-х и т.д.

Суммарное количество передаваемой информации определяется, как $\log_2 n$.

Количество информации для каждого варианта определяется как $\frac{1}{n}\log_2 n$.

Если вероятности для каждого варианта или элемента не являются равными и i-й вариант имеет вероятность P(i), то информация, передаваемая этим вариантом, определяется как $p(i) * \log_2 \frac{1}{p(i)}$.

Количество информации по всем вариантам является их суммой по всем вариантам выражения, которое при равновероятных вариантах сводится к первому выражению.

В результате использования данного способа были сформированы макеты интерфейса и реализованы шаблоны страниц (выполнено в сервисе формирования мокапов интерфейсных страниц https://ninjamock.com/) – рисуноки 2.6 – 2.11.

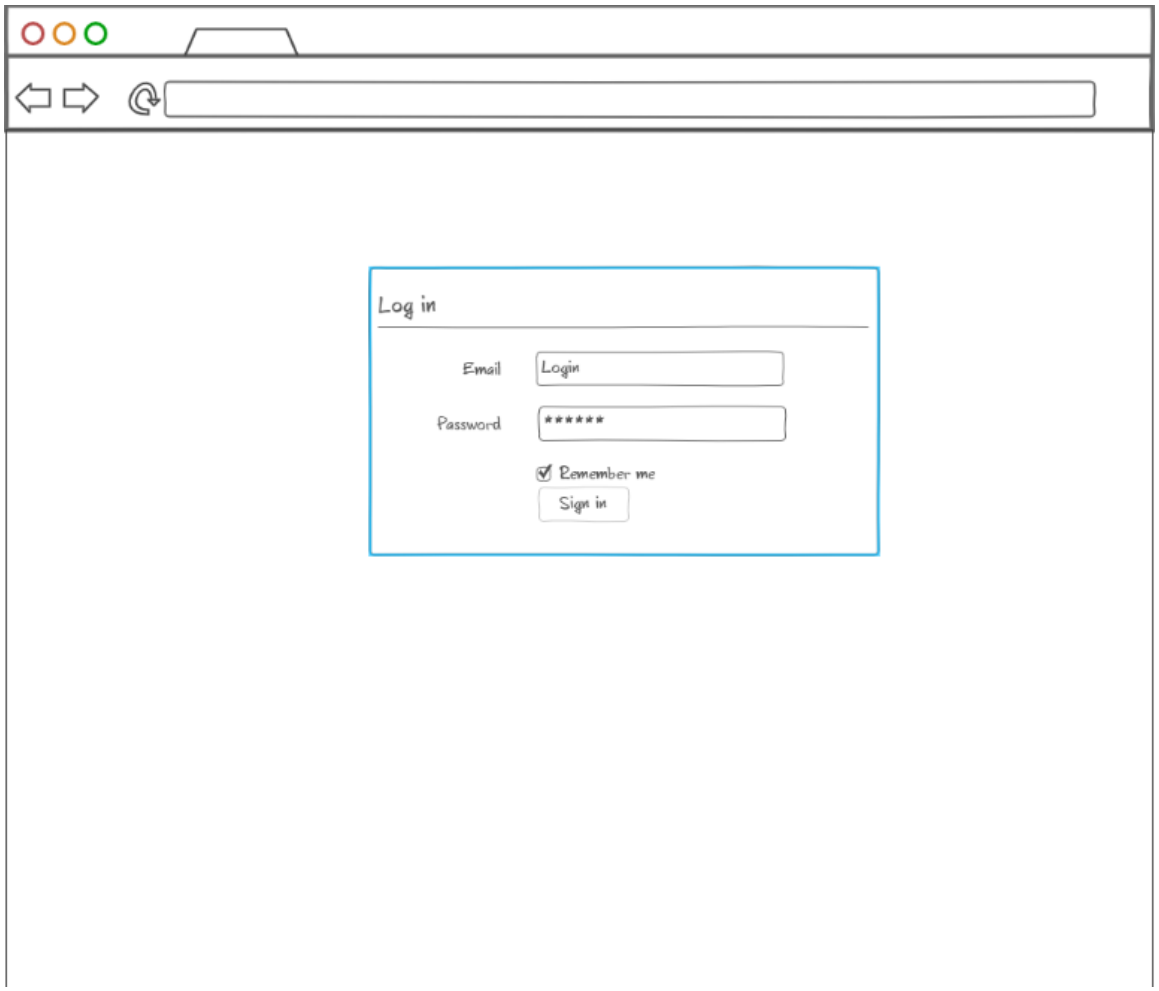


Рисунок 2.6 Шаблон страницы «Логин»

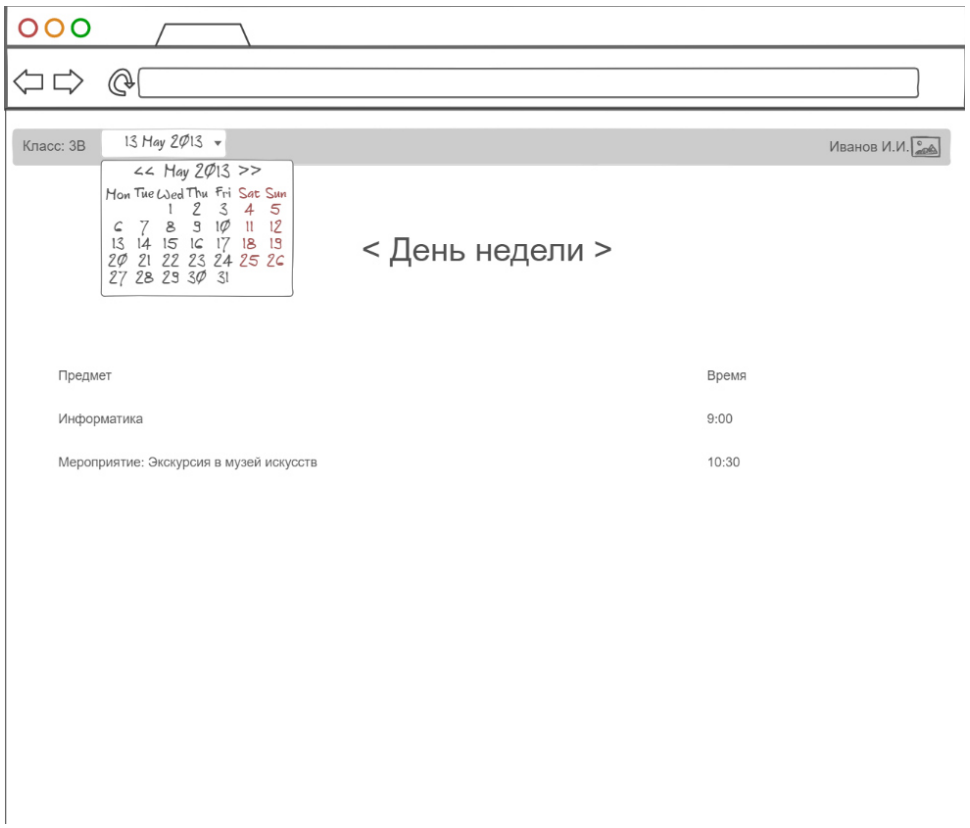


Рисунок 2.7 Шаблон страницы «Ученик»

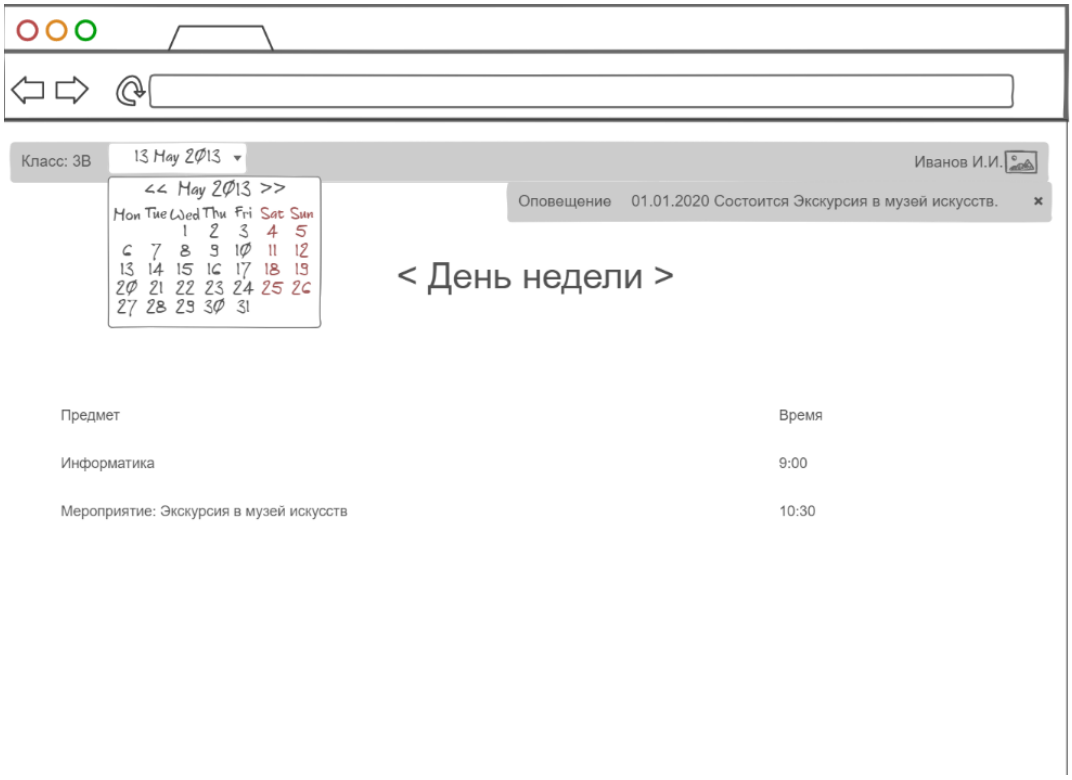


Рисунок 2.8 Шаблон страницы «Ученик» уведомление

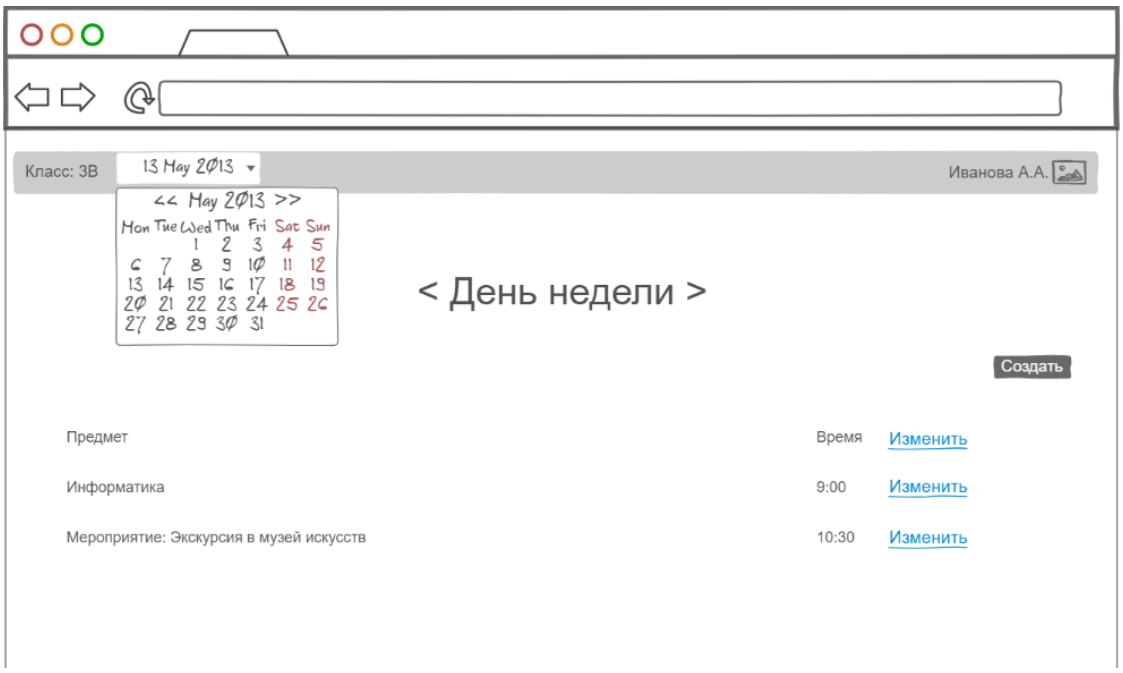


Рисунок 2.9 Шаблон страницы «Учитель»

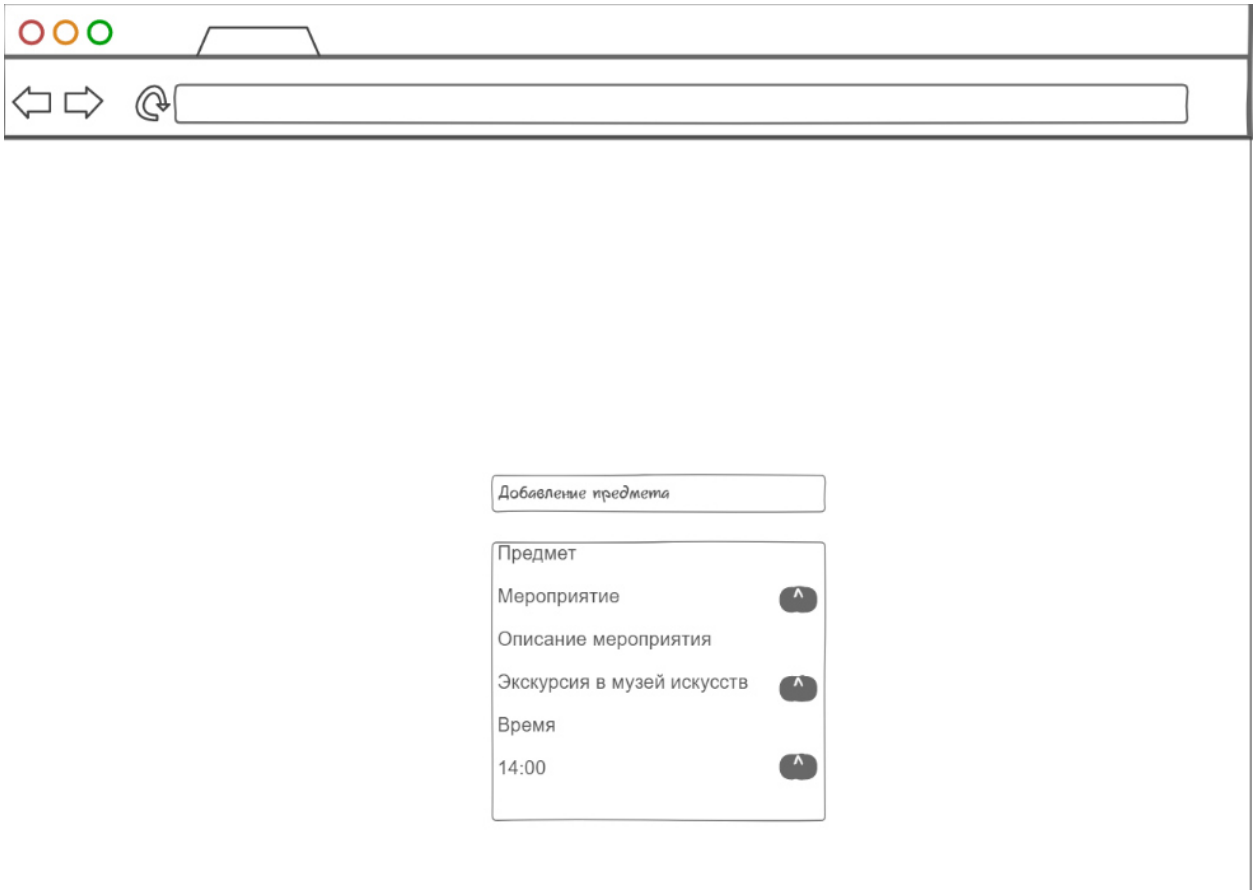


Рисунок 2.10 Шаблон страницы «Добавление мероприятия»

Рисунок 2.11 Шаблон страницы «Добавление предмета»

2.7 Описание работы приложения

После запуска web приложения [localhost] открывается страница аутентификации, в котором пользователь вводит логин и пароль (рисунок 2.12).

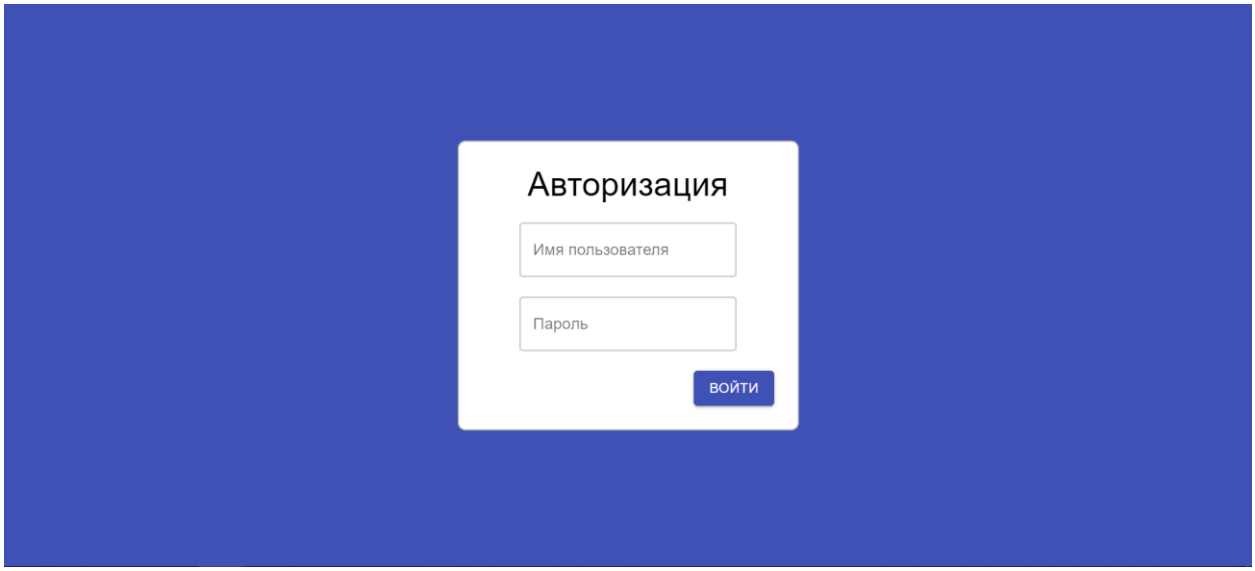


Рисунок 2.12 – Форма аутентификации.

Когда пользователь вводит свои данные (ученик), то он переходит на свою страницу, где может просматривать расписание и мероприятия, которые отсортированы по времени.(рисунок 2.14) Он может выбирать любую дату, кроме воскресенья, потому что в этот день нет занятий. Также, он видит свой класс и имя своего пользователя. Также ученик получает уведомления, когда и во сколько у него будет занятие или мероприятие. (рисунок 2.13).

Вход в систему под пользователем «Учитель» видно расписание и список мероприятий по времени, а также возможность редактировать их в любое время (рисунок 2.15).

Когда учитель редактирует или добавляет занятия или мероприятия у него открывается диалоговое окно, в котором он указывает наименование мероприятия или предмета и назначает время, когда оно будет проходить (рисунки 2.16-17).

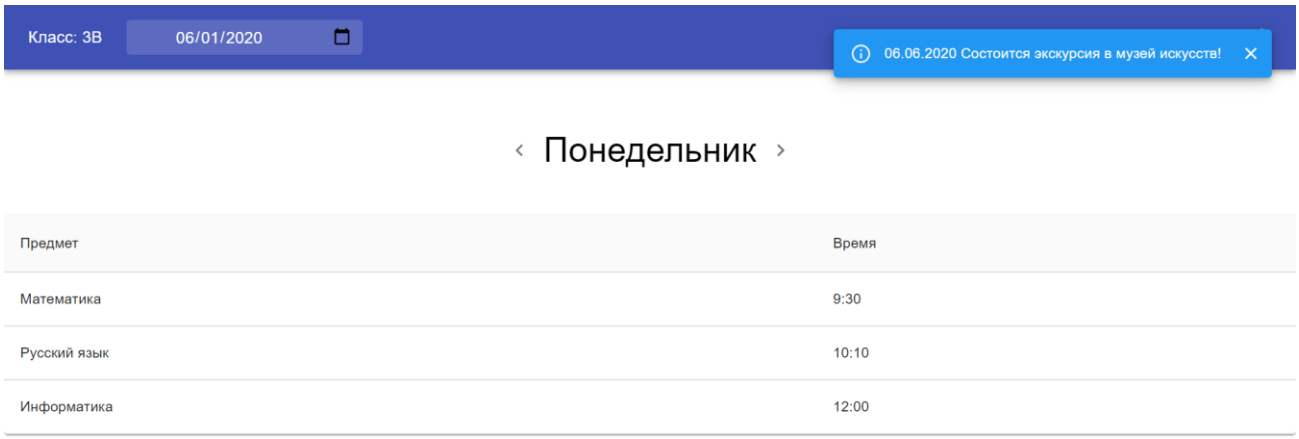


Рисунок 2.13 – Страница ученика с оповещением

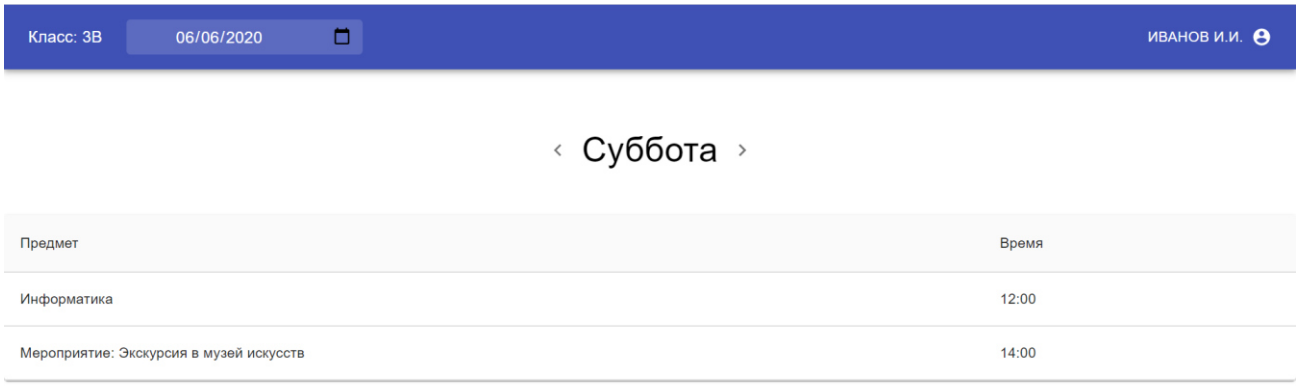


Рисунок 2.14 – Страница ученика

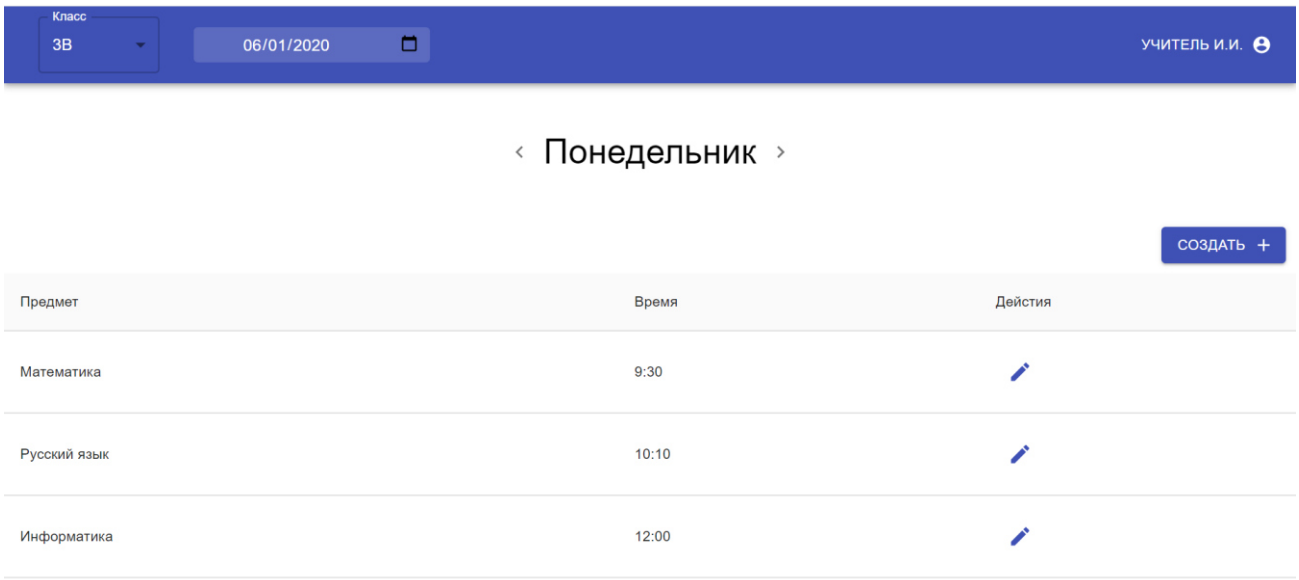


Рисунок 2.15 – Страница учителя

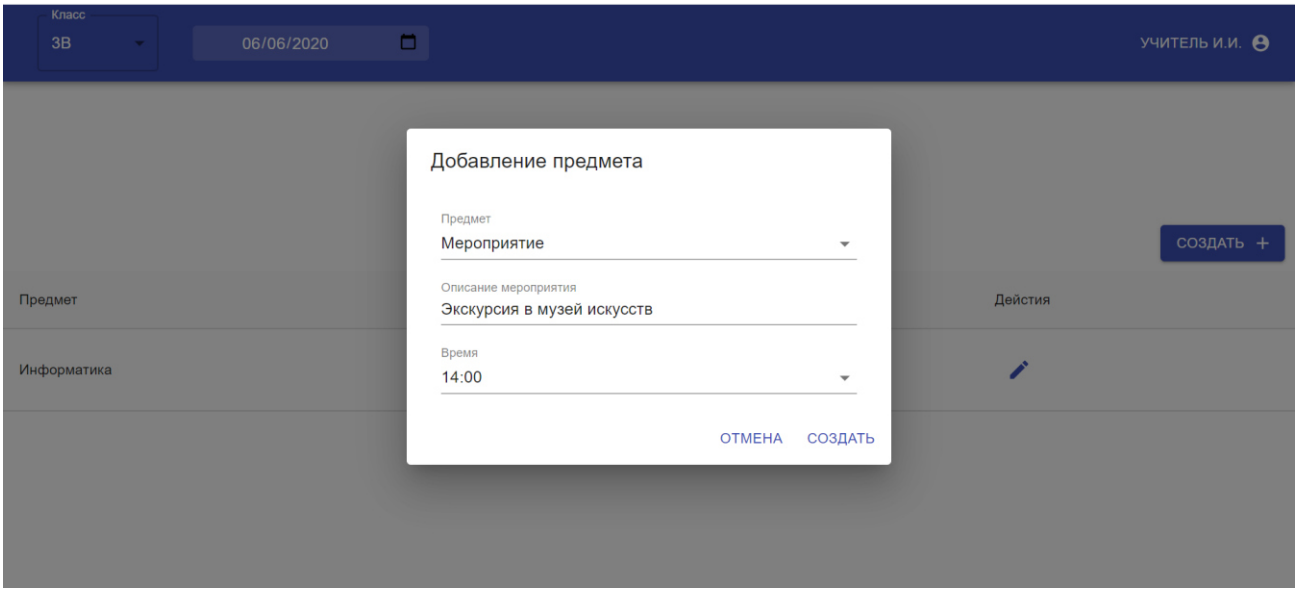


Рисунок 2.16 – Страница добавления мероприятия

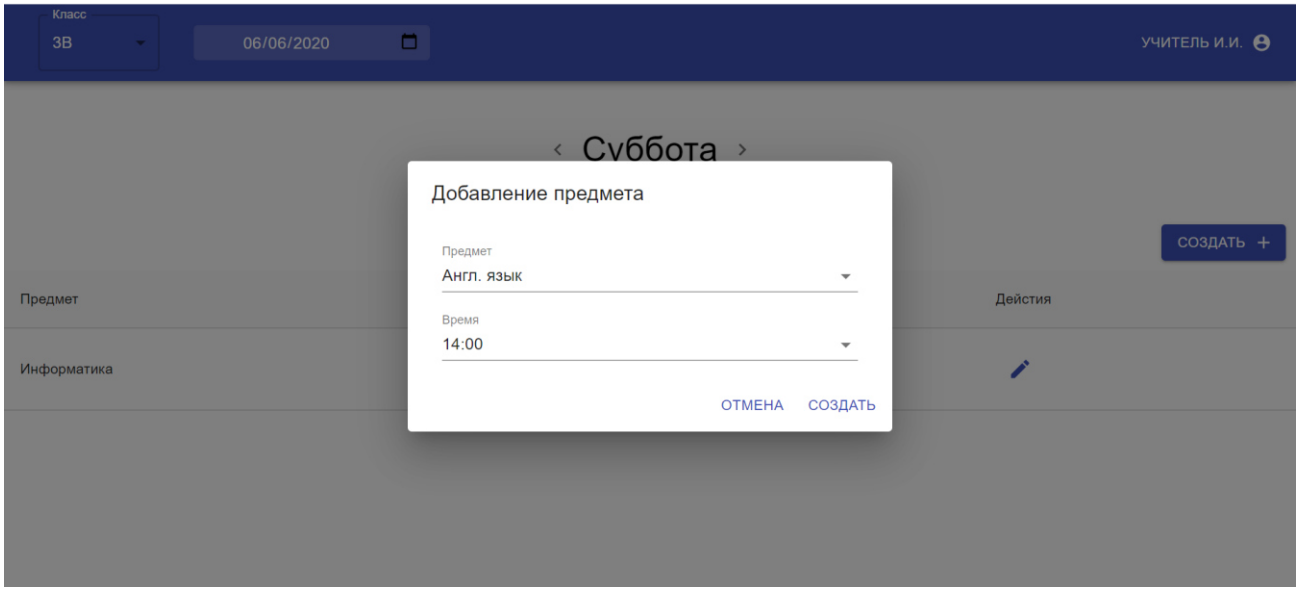


Рисунок 2.17 – Страница добавление предмета

3 Тестирование и развертка программного обеспечения

Для реализации данного дипломного проекта была выбрана среда разработки NetBeans.

Редактор PHP в NetBeans предоставляет шаблоны кода и средства создания кода, используются средства создания методов getter и setter, рефакторинг, подсказки для параметров, советы и функции быстрого исправления. Используется синтаксический и семантический подсказки для написания кода. Существуют такие операторы как try и catch, которые завершают код и имеет автозаполнение параметров метода.

Поддерживает PHP 5.4, тем самым предлагает много функций, связанных с разработкой на PHP 5.4. Использует множество платформ, такие как:

- Zend;
- Symfony2;
- Symfony1.

Поддерживается итерационная разработка, поэтому тестирование PHP проектов выполняется по стандартным шаблонам.

Также существует отладчик кода PHP, который называется Xdebug. Позволяется просматривать код и устанавливать точки останова, где можно найти ошибку в реальном времени. Также IDE NetBeans для PHP поддерживает отладку командной строки: выходные данные отображаются в командной строке в самой среде IDE и пользователи могут проверить созданный без перехода в браузер.

3.1 Создание проекта в NetBeans

Ниже описана технология создания проекта.

Для начала разработки PHP в IDE Netbeans для PHP сначала необходимо создать проект. Проект содержит информацию о размещении файлов проекта и способе запуска и отладки приложения.

Первый шаг, запуск среды IDE, где приступаем к созданию проекта – **File > New > Project > PHP > PHP Aplication**

Выбирается из списка категорий – PHP. В области «Проекты» выбирается «Приложение PHP» и нажимается кнопка далее. Открывается новое окно, где выбирается новое имя и местоположение проекта.

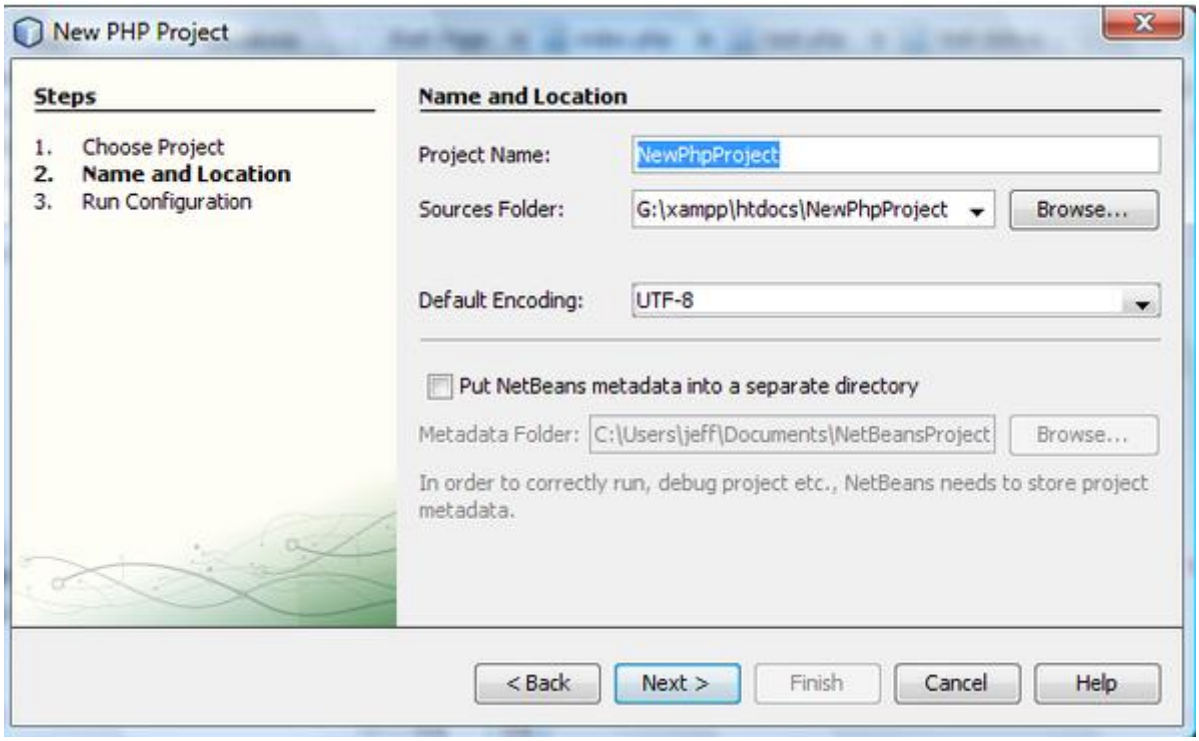


Рисунок 3.1 – Создание нового проекта

В текстовое поле вводится наименование прокта, в нашем случае называется NewPHPProject.

В поле исходной папки перейти к корню документов PHP и создается подпапка NewPHPProject. Корень документов – эта папка, в которой веб-сервер ищет файлы для открытия в браузере. Корневой узел документов

указан в файле настройки веб-сервера. Например, в Xampp корнем документов является папка XAMPP_HOME/htdocs.

В остальных полях оставляется значение по умолчанию и нажимается «Далее» и открывается окно «Настройки выполнения» (рисунок 3.2).

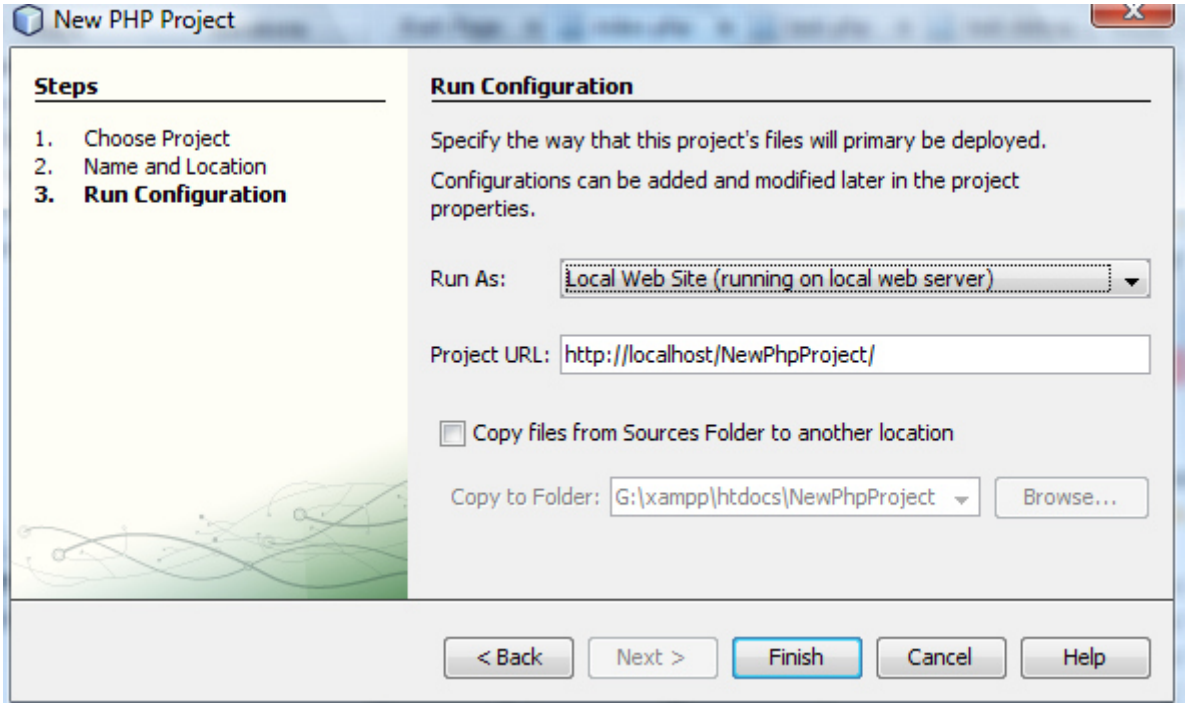


Рисунок 3.2 – Окно «Настройка выполнения»

В раскрывающемся списке «Выполнить как» выберается «Локальный веб-сайт». Начнется выполнение проекта на локальном сервере Apache. Проект можно также выполнить удаленно через FTP или запустить его из командной строки. Поле «URL-адрес проекта» остается без изменений. Завершается и создается проект с помощью среды IDE.

В результате вышеперечисленных операций появляется окно нового проекта, в котором присутствует структура каталога проекта и сгенерированные автоматически файлы настройки (рис. 3.3).

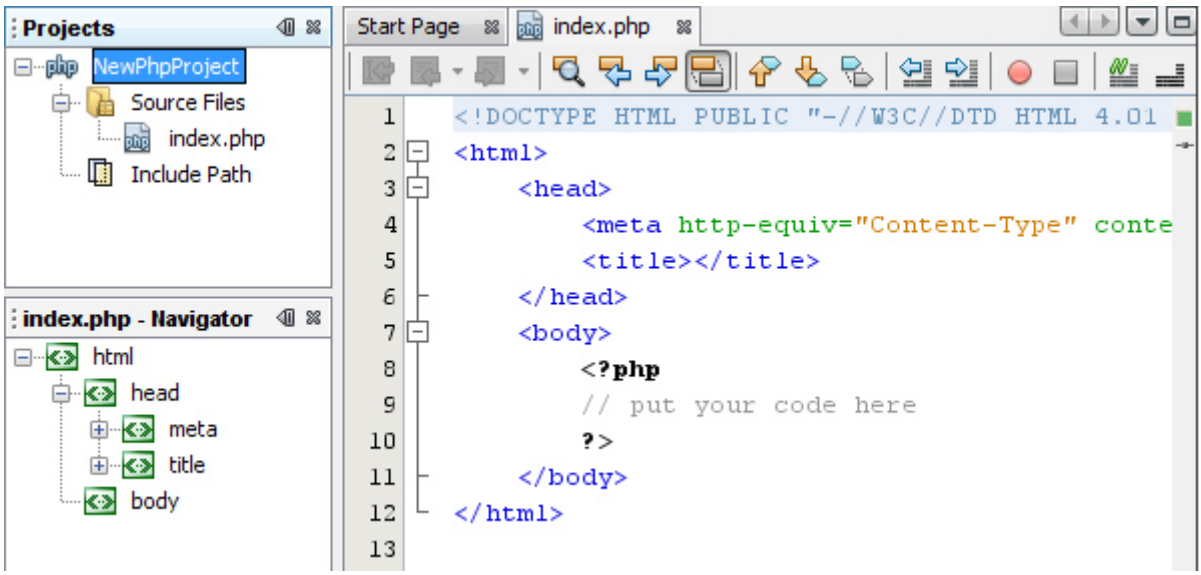


Рисунок 3.4 – Окно нового проекта

По умолчанию в проект добавлен файл index.php, который является главным файлом проекта, и к которому будут подключаться остальные файлы.

Для добавления новых файлов в проект нужно воспользоваться пунктом меню - **File > New > New File.**

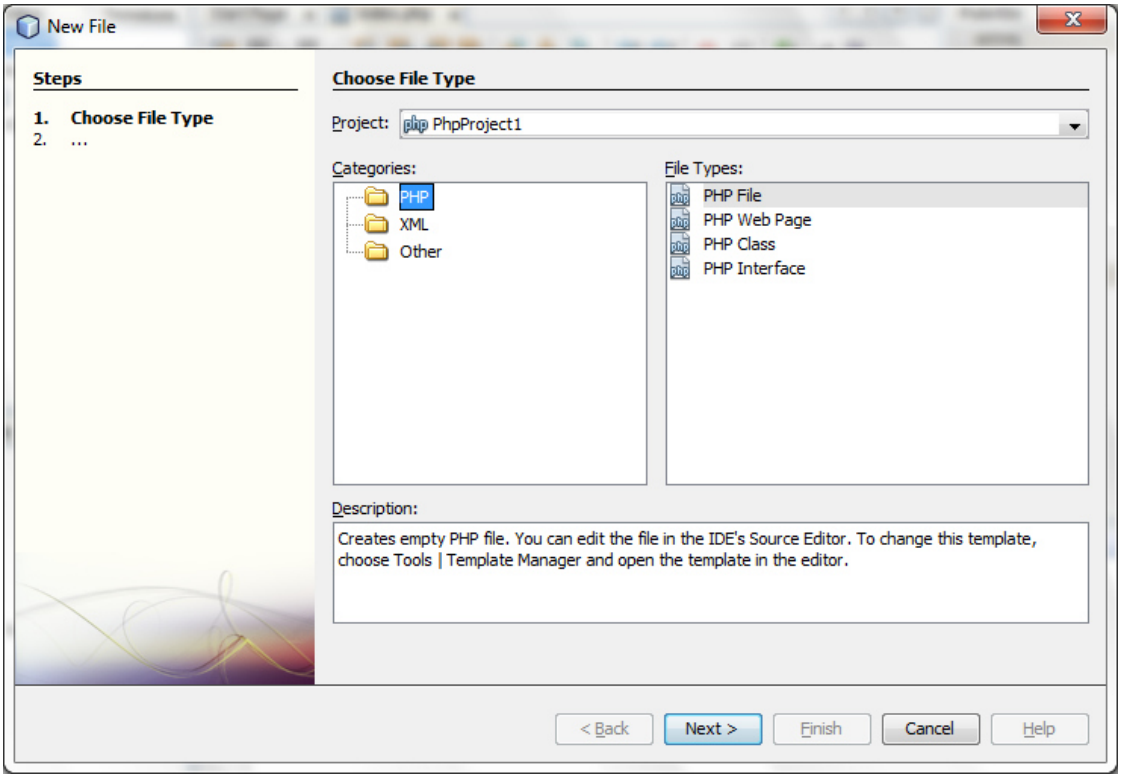


Рисунок 3.5 – Добавление нового файла в проект

После того как выбрали тип файла, который нужно добавить в проект, следует нажать кнопку «Next» для того чтобы задать имя файлу.

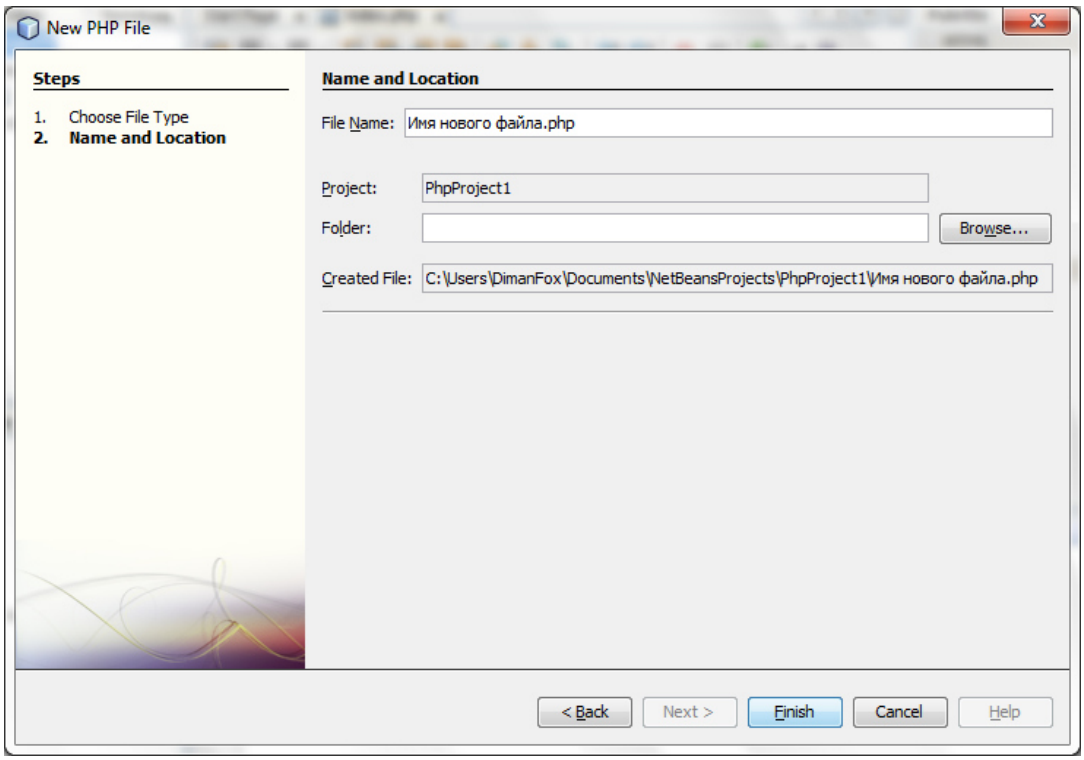


Рисунок 3.6 – Добавление нового файла в проект

После того как задали имя новому файлу, можно нажать кнопку «Finish» и новый файл добавиться в проект с минимум кода-шаблоном.

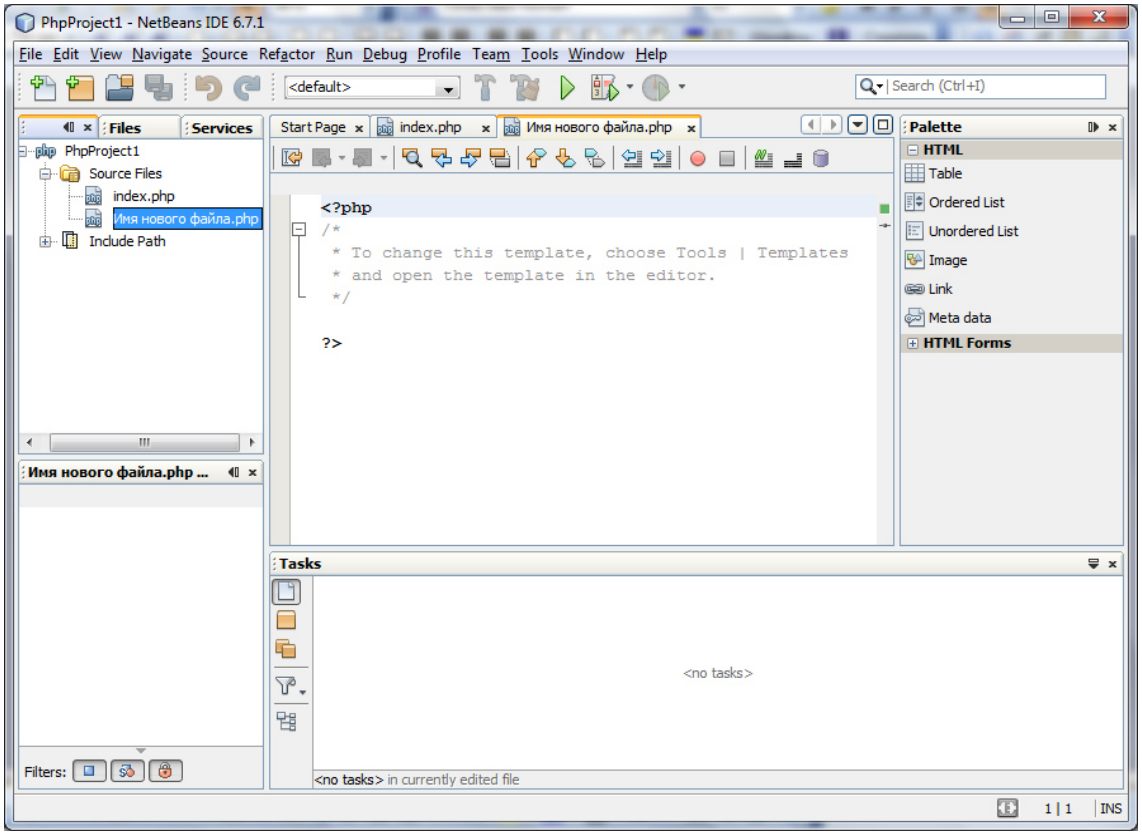


Рисунок 3.7 – Окно проекта

3.2 Отладка кода с помощью XDebug

Отладка программного обеспечения самая ненавистная работа доля разработчика. Большинство используют для отладки связку echo(print_r,var_dump) и exit(die), переходя от одной строке к другой. Однако, если ошибка появляется вновь в этом файле, требуется заново прописывать отладочные команды. Можно конечно отладку обернуть внутри конструкции if, которая будет работать только, когда определена какая-либо константа, например DEBUG. Но каждый такой if будет слегка замедлять производительность, да и смотреться будет в коде очень ужасно.

Xdebug позволяет создавать логи трассировки, достаточно хороший выход для данной ситуации, и не надо будет изменять программу. Однако лог трассировки, особенно когда он создан для части программы, предоставляет нам много информации, не имеющей никакого отношения к отладке, таким образом, использование отладчика это оптимальное решение. Отладчик позволит вам остановить запуск программы на некоторое время, проверить или модифицировать текущее значение переменной, а затем продолжить выполнение программы. Запуская программу шаг за шагом, вы можете близко взглянуть как выполняется код, что поможет найти места ошибок.

В среде разработки NetBeans IDE для отладки PHP кода есть очень удобное решение – Xdebug. Для его настройки необходимо:

В соответствии с версией php, выбрать и скачать библиотеку xdebug .

Скопировать файл php_xdebug.dll в папку /php/ext/.

В файле php.ini добавить секцию XDebug и прописать в ней сладущее:

```
[XDebug]
extension=php_xdebug-2.0.5-5.2.dll
zend_extension_ts="C:\server\php\ext\php_xdebug-2.0.5-5.2.dll"
xdebug.remote_enable=on
xdebug.remote_handler=dbgp
```

xdebug.remote_host=localhost

xdebug.remote_port=9000

Перезапускаем Apache

В NetBeans IDE перейти в Tools-Options-PHP. На вкладке General в секции Command Line указать путь до php.exe. В секции Debugging прописать port 9000 и Session ID netbeans-xdebug.

3.2.1 Как работает отладка

Когда отладка включена в php.ini, xdebug начинает контролировать выполнение программы, это значит, что xdebug может останавливать выполнение программы в любой момент, а потом запускать ее сново.

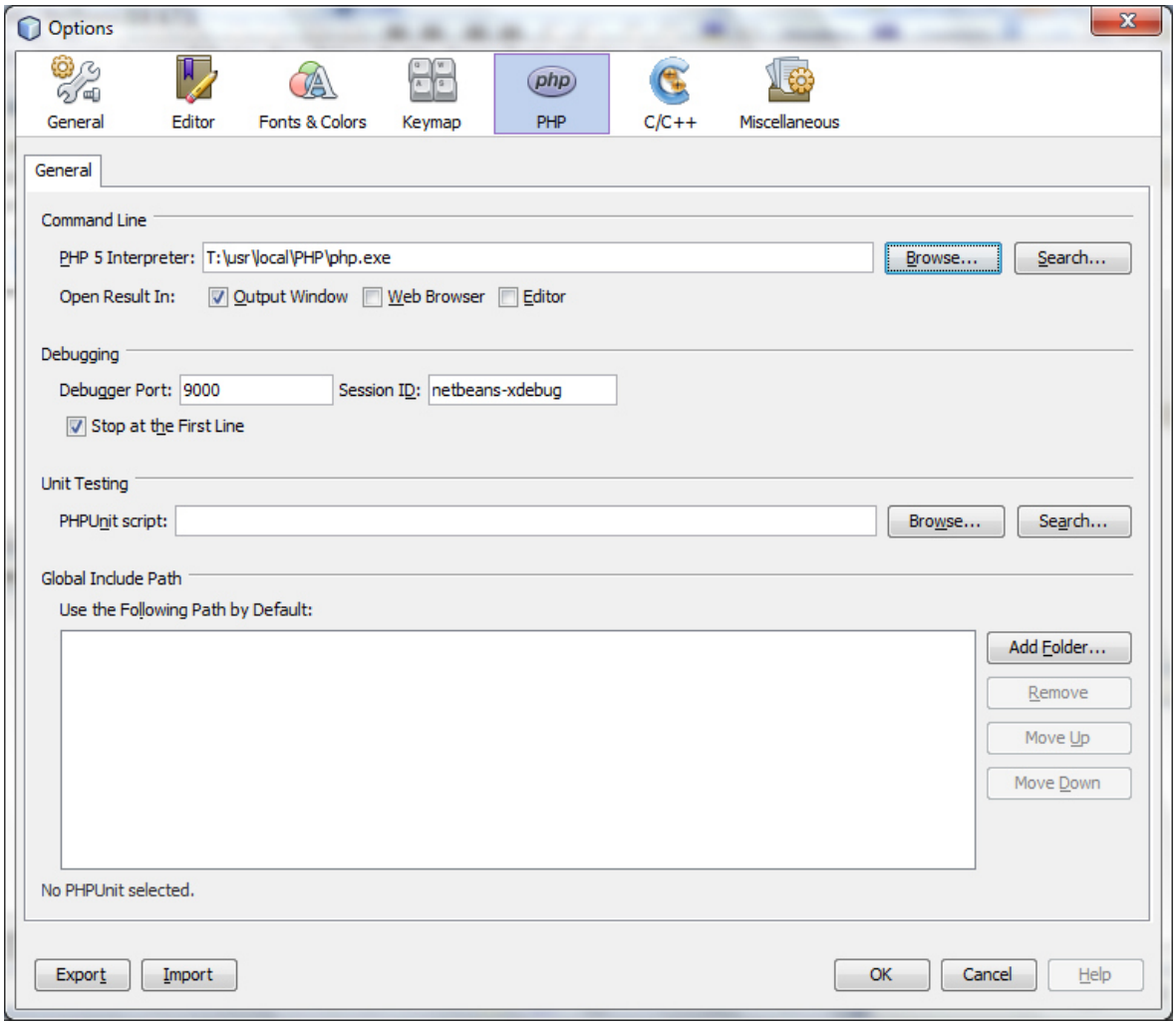


Рисунок 3.8 – Настройка отладчика

Когда выполнение программы остановлено, xdebug может получить информацию о текущем состоянии программы, например, прочитать текущие значения переменных. Также xdebug представляет возможность изменять значения переменных во время выполнения скрипта.

Расширение xdebug – это сервер, ожидающие соединений с клиентами на определенном порту, задаваемом в конфигурации. Существует два протокола, которые могут быть использованы для коммуникации между xdebug-клиентом и xdebug-сервером (GDB и DBGp).

GDB – это старый протокол, который был заменен на DBGp. Посылая команды серверу-xdebug, xdebug-клиент удаленно контролирует выполнение, сообщает PHP о приостановке запуска, выполнении следующей строчки кода или продолжении выполнения программы. Клиент обычно встроен в редактор или IDE (в нашем случае в NetBeans), поэтому вы не будете иметь дело с протоколом общения напрямую.

Веб-сервер с xdebug может быть запущен на другой операционной системе, нежели xdebug-клиент. Вот почему xdebug называется удаленным отладчиком(remote debugger). Для упрощения, мы настроим сервер и клиент на одном и том же компьютере.

Существует два режима начала сессии отладки. Эти режимы контролируются в php.ini настройкой xdebug.remote_mode. Значение по умолчанию req, xdebug-клиент будет соединятся с сервером всякий раз когда начинается выполнение скрипта. Если вы хотите, чтобы xdebug соединялся с сервером только тогда, когда установлен breakpoint или когда в скрипте возникла ошибка, то можно установить xdebug.remote_mode в jit.

Для того чтобы начать отладку, нужно послать параметр XDEBUG_SESSION_START в скрипт как часть GET, POST или COOKIE. Значение этого параметра это название сессии отладки, которое должно быть уникальным, по имени сессии xdebug может различать различные сессии, запущенные параллельно. Для завершения сессии отладки необходимо послать скрипту команду XDEBUG_SESSION_STOP.

Вместо того чтобы вручную начинать и заканчивать отладочные сессии, следует установить специальный firefox plugin, который поможет легко начинать и заканчивать сессии одним нажатием мышки.

4. Технико-экономическое обоснование разработки программного обеспечения дипломного проекта

4.1 Организация создания программного обеспечения

Целью данного дипломного проекта, является создание программных средств автоматизированной системы контроля расписания.

Программу, как любое техническое решение, необходимо рассматривать с точки зрения экономической целесообразности и пользы.

Программное изделие должно быть разработано так, чтобы оно выполняло свои функции без лишних затрат ресурсов (оперативной памяти ЭВМ, машинного времени, пропускной способности каналов передачи данных и др. –

на стадии функционирования; время разработки и денежных ресурсов – на стадии использования программного изделия).

Содержание технико-экономического обоснования разработки программы заключается в следующем:

1) расчет технико-экономических показателей;
2) расчет затрат на разработку;

Приведенные в данном разделе дипломной работы данных позволяют сопоставить результаты разработки и затраты на нее, чтобы сделать вывод об эффективности проекта.

4.2 Расчет цены программного обеспечения

Расчет цены ПО содержит:

1) определение затрат на оплату труда и накладных расходов;
2) установление цены ПО;
3) установление нижнего предела цены и договорной цены с учетом тиражирования.

4.2.1 Определение затрат на разработку программного обеспечения

Затраты на разработку программного обеспечения определяются по формуле:

$$З_{р.по} = З_{р} + З_{д} + З_{с} + З_{н} + Р_{экспл}, \quad (4.1)$$

где $З_р$ – заработная плата разработчиков, руб.;

$З_д$ – дополнительная заработная плата, руб.;

$З_с$ – отчисления на социальные нужды, руб.;

$З_н$ – накладные расходы, руб.;

$Р_{экспл}$ – эксплуатационные расходы, руб.

Оплата труда разработчика 8000 руб. в месяц, с учетом того, что в месяце 24 рабочих дня, получим 333 руб. в день. В общей сложности, разработка ПО велась в течение 40 рабочих дней. Таким образом, общая заработная плата за 40 дней разработки составит

$$З_р = 40 \cdot 333 = 13320 \ (руб.)$$

Дополнительная заработная плата определяется по формуле:

$$З_д = К_д \cdot З_р, \quad (4.2)$$

где $К_д$ – коэффициент, учитывающий дополнительную заработную плату.

Примем $К_д$ равным 0,1, тогда получаем, что дополнительная заработная плата составляет:

$$З_д = 13320 \cdot 0,1 = 1332 \ (руб.)$$

Расчет отчислений в страховые фонды с фонда заработной платы всех категорий промышленно-производственного персонала проводится в соответствии с установленным по действующему законодательству значением процента отчислений определяются по формуле:

$$З_с = (З_р + З_д) \cdot 0,30 \quad (4.3)$$

Учитывая ранее полученные значения заработной платы разработчика и дополнительной заработной платы, отчисления на социальные нужды составляют:

$$З_с = (13320 + 1332) \cdot 0{,}30 = 4395{,}6\ (руб.)$$

Накладные расходы определяются следующим образом:

$$З_н = \frac{З_р \cdot П}{100}, \qquad (4.4)$$

где П – процент накладных расходов организации, в которой разрабатывается программа.

Примем, что П равен 45 %, получим накладные расходы в размере

$$\boldsymbol{З_н = \frac{13320 \cdot 45}{100}} = 5994\ (\boldsymbol{руб.})$$

Расчет стоимости одного часа эксплуатации ЭВМ производится по формуле:

$$Р_{экспл} = \frac{С_э + С_о + А + С_р}{Ф_г}, \qquad (4.5)$$

где $С_э$ – стоимость энергии, потребляемой компьютером за год, руб.;

$С_о$ – расходы на обслуживание компьютера в течение года, руб.;

А – амортизационные отчисления, руб.;

$С_р$ – расходы на текущий ремонт, руб.;

$Ф_г$ – годовой полезный фонд времени работы ЭВМ, ч.

Компьютер работает в течение 9 часов в сутки, пять дней в неделю, 249 дней в году. Годовой полезный фонд времени работы компьютера при таких условиях составляет

$$Ф_г = 9 \cdot 249 = 2241\ (ч.)$$

Стоимость потребляемой за год электроэнергии рассчитывается по формуле:

$$С_э = М_п \cdot Ф_г \cdot Ц_э, \qquad (4.6)$$

где $M_п$ – потребляемая мощность ЭВМ, кВт;

$Ц_э$ – цена 1 кВт*ч электроэнергии, руб.

Согласно последней тарифной ставке 1 кВт*ч электроэнергии стоит 3,44 руб. Потребляемая мощность компьютера 0,2 кВт. Затраты на электроэнергию составят

$$С_э = 0,2 \cdot 2241 \cdot 3,44 = 1541,81 \ (руб.)$$

Расходы на обслуживание компьютера за год определяются по формуле:

$$С_о = \frac{ЗП_о \cdot 12}{Э_о}, \quad (4.7)$$

где $ЗП_о$ – зарплата персонала, обслуживающего компьютеры, руб.;

$Э_о$ – количество обслуживаемых им единиц оборудования;

12 – количество месяцев в году.

В организации используется 5 компьютеров, обслуживанием которых занимается 1 человек. Его заработная плата составляет 5000 руб. в месяц. В соответствии с формулой (4.7), затраты на обслуживание ЭВМ составляют

$$С_о = \frac{5000 \cdot 12}{5} = 12000 \ (руб.)$$

Амортизационные отчисления определяются по формуле:

$$А = \frac{Н_а \cdot С}{100}, \quad (4.8)$$

где $Н_а$ – норма амортизации. Примем ускоренную, равную 50 %;

С – балансовая стоимость компьютера, руб.

Балансовая стоимость складывается из стоимости компьютера, стоимости транспортировки и стоимости установки. Для используемого компьютера балансовая стоимость составляет 15000 руб.

Таким образом, согласно формуле (4.8), амортизационные отчисления составят:

$$А = \frac{50 \cdot 15000}{100} = 7500 \ (руб.)$$

Расходы на текущий ремонт $С_р$ определяются как 5% от балансовой стоимости компьютера. Таким образом, получаем, что расходы составят:

$$С_р = 0,05 \cdot 15000 = 750 \ (руб.)$$

Подставляя вычисленные значения в формулу (4.5) получаем, что стоимость одного часа эксплуатации ЭВМ равна

$$Р_{экспл/ч} = \frac{1541,81 + 12000 + 7500 + 750}{2241} = 9,72\ (руб.)$$

Из 40 дней, затраченных на разработку ПО, 28 дней приходится на кодирование и 12 дней на отладку ПО. Учитывая, что в день компьютер работает в течение 9 часов, получаем, что эксплуатационные расходы составят:

$$Р_{экспл} = (28 + 12) \cdot 9 \cdot 9,72 = 3499,2\ (руб.)$$

Согласно формуле (3.1) затраты на разработку программного обеспечения составят:

$$З_{р.по} = 13320 + 1332 + 4395,6 + 5994 + 3499,2 = 28540,8$$ (*руб.)*

4.2.2 Установление цены программного обеспечения

Цена разработанной программы определяется по формуле:

$$\boldsymbol{Ц = З_{р.по} \cdot (1 + Р),} \qquad (4.9)$$

где Р – норматив рентабельности, позволяющий учитывать прибыль организации, разрабатывающей данную программу.

Примем норматив рентабельности равным 20%, тогда цена разрабатываемого программного продукта:

$$Ц = 28540,8 \cdot 1,2 = 34248,96 (руб.)$$

Итак, цена программного продукта, полученного в ходе дипломного проектирования, составляет 34248,96 руб. при норме рентабельности 20%. Себестоимость программы – 28540,8 руб.

4.2.3 Установление нижнего предела цены с учетом тиражирования

Нижний предел цены (Цнп) с учетом желаемой рентабельности (Р), тиражированием (N) и НДС (Н) определяется по формуле:

$$Ц_{НП} = \frac{З_{р.по} \cdot (1 + Р)}{N} \cdot (1 + Н). \quad (4.10)$$

При установленном НДС равным 20%, и учитывая, что тиражирование программы не предполагается, рассчитываем предел нижней цены.

$$Ц_{НП} = \frac{34248,96 \cdot (1 + 0,20)}{1} \cdot (1 + 0,20) = 49318,5 \ (руб.)$$

Окончательную цену разрабатываемого ПО примем на уровне 49318,5 рублей.

5 Безопасность жизнедеятельности

5.1 Организация безопасных условий труда на рабочем месте оператора ПЭВМ

Рабочее место - это система функционально и пространственно организованных технических средств и предмета труда, обеспечивающая условия для успешного решения человеком - оператором поставленной перед ним задачи.

На комфортность работы оператора влияет организация рабочего места оператора, средства отображения и органы управления машиной. Они должны быть максимально удобны для человека, чтобы не создавать помех, дискомфорта в процессе работы, а также способствовать наименьшей утомляемости.

Основным способом обеспечения условий комфорта оператора ПЭВМ является рациональная организация его рабочего места.

Под рабочим местом оператора ПЭВМ понимается зона трудовой деятельности в системе «человек-машина» с техническими средствами и вспомогательным оборудованием, необходимым для решения конкретных производственных задач.

Конструкция рабочего стола должна обеспечивать оптимальное размещение на рабочей поверхности используемого оборудования с учетом его количества и конструктивных особенностей, характера выполняемой работы. При этом допускается использование рабочих столов различных конструкций, отвечающих современным требованиям эргономики. Поверхность рабочего стола должна иметь коэффициент отражения 0,5 - 0,7. [1]

Рекомендуемые расположения текста, клавиатуры и экрана компьютера R=50..60 см для типовых компьютеров с размером экрана по диагонали 250..350 мм (предельно минимальное) представлены на рисунке 5.1.

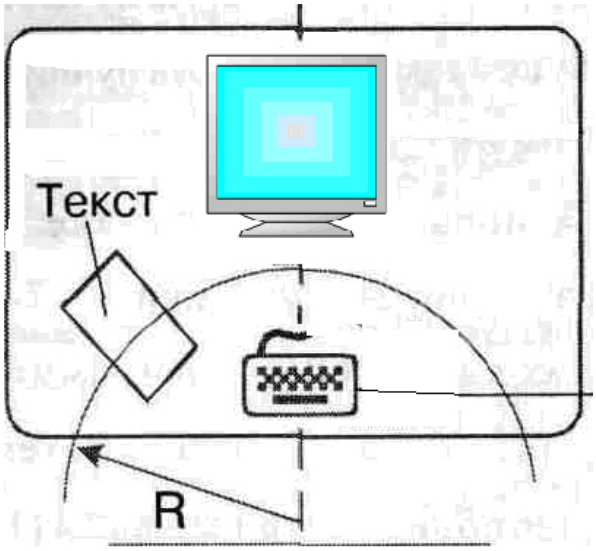


Рисунок 5.1- Рабочее пространство

Конструкция рабочего стула (кресла) должна обеспечивать поддержание рациональной рабочей позы при работе на ПЭВМ позволять изменять позу с целью снижения статического напряжения мышц шейно-плечевой области и спины для предупреждения развития утомления. Тип рабочего стула (кресла) следует выбирать с учетом роста пользователя, характера и продолжительности работы с ПЭВМ.

Рабочий стул (кресло) должен быть подъемно-поворотным, регулируемым по высоте и углам наклона сиденья и спинки, а также расстоянию спинки от переднего края сиденья, при этом регулировка каждого параметра должна быть независимой, легко осуществляемой и иметь надежную фиксацию.

Поверхность сиденья, спинки и других элементов стула (кресла) должна быть полумягкой, с нескользящим, слабо электризующимся и воздухопроницаемым покрытием, обеспечивающим легкую очистку от загрязнений.

Высота рабочей поверхности стола для взрослых пользователей должна регулироваться в пределах 680 - 800 мм; при отсутствии такой возможности высота рабочей поверхности стола должна составлять 725 мм.

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

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.

Конструкция рабочего стула должна обеспечивать:

- ширину и глубину поверхности сиденья не менее 400 мм;

- поверхность сиденья с закругленным передним краем;

- регулировку высоты поверхности сиденья в пределах 400 - 550 мм и углам наклона вперед до 15 градусов, и назад до 5 градусов;

- высоту опорной поверхности спинки 300 ±20 мм, ширину - не менее 380 мм и радиус кривизны горизонтальной плоскости - 400 мм;

- угол наклона спинки в вертикальной плоскости в пределах ±30 градусов;

- регулировку расстояния спинки от переднего края сиденья в пределах 260 - 400 мм;

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

- регулировку подлокотников по высоте над сиденьем в пределах 230 ±30 мм и внутреннего расстояния между подлокотниками в пределах 350 - 500 мм.

Параметры рабочего стула (кресла) представлены на рисунке 5.2.

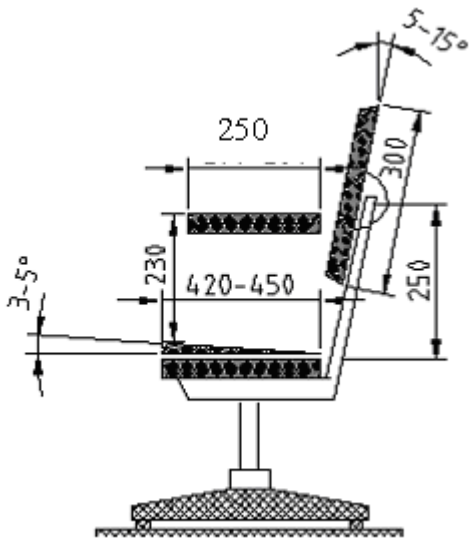


Рисунок 5.2 – Рабочий стул

Рабочее место пользователя ПЭВМ следует оборудовать 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°. Поверхность подставки должна быть рифленой и иметь по переднему краю бортик высотой 10 мм.

Клавиатуру следует располагать на поверхности стола на расстоянии 100 - 300 мм от края, обращенного к пользователю или на специальной, регулируемой по высоте рабочей поверхности, отделенной от основной столешницы.

Экран должен находиться в центре поля обзора, примерно на расстоянии 60 - 70 см от глаз, при этом смотреть на него следует сверху вниз, учебники, тетради должны находиться слева на столе. Положение тела оператора относительно экрана должно соответствовать направлению просмотра экрана компьютера под прямым углом или под углом 75 градусов.[2] Рекомендации по рациональному размещению средств отображения и информации представлены на рисунке 5.3:

1 - возможный наклон плоскости для представления информации,

2 - границы полезной зоны,

3 - граница информации,

4 - горизонталь,

5 - граница оптимальной зоны,

6 - нормальная линия взгляда,

7 - граница оптимальной зоны,

8 - возможный наклон плоскости для представления информации,

9 - граница информации,

10 - граница полезной зоны.

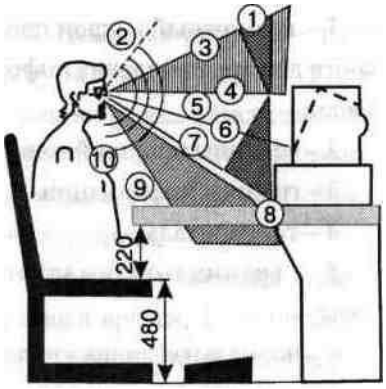


Рисунок 5.3 – Размещение средств отображения и информации

ПЭВМ должны соответствовать требованиям санитарных правил, и каждый их тип подлежит санитарно-эпидемиологической экспертизе с оценкой в испытательных лабораториях, аккредитованных в установленном порядке.

Допустимые значения напряженности электрического поля в диапазоне частот 5 Гц – 2 кГц составляет 25 В/м, в диапазоне частот 2 кГц – 400 кГц составляет 2,5 В/м., напряженность электрического поля – 15 кВ/м.

Концентрации вредных веществ, выделяемых ПЭВМ в воздух помещений, не должны превышать предельно допустимых концентраций (ПДК), установленных для атмосферного воздуха.

Мощность экспозиционной дозы мягкого рентгеновского излучения в любой точке на расстоянии 0,05м от экрана и корпуса монитра при любых положениях регулировочных устройств не должна превышать 1 мкЗв/час (100 мкР/час).

Конструкция ПЭВМ должна обеспечивать возможность поворота корпуса в горизонтальной и вертикальной плоскости с фиксацией в заданном положении для обеспечения фронтального наблюдения экрана монитора. Дизайн ПЭВМ должен предусматривать окраску корпуса в спокойные мягкие тона с диффузным рассеиванием света. Корпус ПЭВМ, клавиатура и другие блоки и устройства ПЭВМ должны иметь матовую поверхность с коэффициентом отражения 0,4-0,6 и не иметь блестящих деталей.

Документация на проектирование, изготовление и эксплуатацию ПЭВМ не должна противоречить требованиям санитарных правил.

Рабочие столы следует размещать таким образом, чтобы мониторы были ориентированы боковой стороной к световым проемам, чтобы естественный свет падал преимущественно слева.

Искусственное освещение в помещениях для эксплуатации ПЭВМ должно осуществляться системой общего равномерного освещения. В производственных и административно-общественных помещениях, в случаях преимущественной работы с документами, следует применять системы комбинированного освещения (к общему освещению дополнительно устанавливаются светильники местного освещения, предназначенные для освещения зоны расположения документов).

Освещенность на поверхности стола в зоне размещения рабочего документа должна быть 300-500лк. Освещение не должно создавать бликов на поверхности экрана. Освещенность поверхности экрана не должна быть более 300лк.

В качестве источников света при искусственном освещении следует применять преимущественно люминесцентные лампы типа ЛБ и компактные люминесцентные лампы (КЛЛ).

Коэффициент запаса (Кз) для осветительных установок общего освещения должен приниматься равным 1,4.

Коэффициент пульсации не должен превышать 5%.

Для обеспечения нормируемых значений освещенности в помещениях для использования ПЭВМ следует проводить чистку стекол оконных рам и светильников не реже двух раз в год и проводить своевременную замену перегоревших ламп.

Основные и производственные помещения вычислительного центра целесообразно окрашивать в соответствии с цветом технических средств. Выбор цвета определяется рядом факторов, в том числе конструкцией здания, характером выполняемой работы, освещенностью, количеством работающих.

Необходимо учитывать, что цвет является сильным психологическим стимулятором:

красный - увеличивает мускульное напряжение;

оранжевый - стимулирует деятельность;

желтый - стимулирует зрение и нервную систему;

зеленый - успокаивает;

голубой - ослабляет мускульное напряжение и т.д.

Восприятие цвета в большой степени зависит от освещенности. Под влиянием различных источников света цвет поверхности меняет тон.

К системам производственного освещения предъявляются следующие основные требования:

Соответствие уровня освещенности рабочих мест характеру выполняемой зрительной работы.

Достаточно равномерное распределение яркости на рабочих поверхностях в окружающем пространстве.

Отсутствие резких теней, прямой и отражённой блёскости (блёсткость - повышенная яркость светящихся поверхностей, вызывающая ослеплённость).

Постоянство освещённости во времени.

- Оптимальная направленность излучаемого осветительными приборами светового потока.

В вычислительном центре, применяем одностороннее боковое естественное освещение. Причем светопроемы с целью уменьшения солнечной инсоляции устраивают с северной, северо-восточной ориентацией. В машинных залах рабочие места операторов, работающих с дисплеями, располагают подальше от окон и таким образом, чтобы оконные проемы находились сбоку. Если экран дисплея обращен к оконному проему, необходимы специальные экранирующие устройства. Окна рекомендуется снабжать светорассеивающими шторами, регулируемыми жалюзи или солнцезащитной пленкой с металлизированным покрытием.

Проблемы гигиены труда пользователей ПЭВМ относятся к числу наиболее актуальных, поскольку непрерывно расширяется круг задач, решаемых ПЭВМ, и все больше людей вовлекается в процесс использования ВТ. В последние годы Северо-Западным Научным центром гигиены и общественного здоровья проводились комплексные гигиенические исследования по оценке условий труда и состояния здоровья работающих с ПЭВМ. Анализ полученных результатов, позволяет составлять определенное представление о факторах риска здоровью пользователей ПЭВМ.

Рациональный режим труда и отдыха способствует снижению утомляемости, поддержанию высокой работоспособности оператора ПЭВМ, повышению производительности его труда и сохранению здоровья. При установлении количества перерывов на отдых, определении их длительности и распределении их в течении рабочей смены учитывается специфика производства, напряженность труда, изменение работоспособности человека в процессе работы.[5]

В качестве таких мер рекомендуется применение в рамках режима труда и отдыха различные физкультурно-оздоровительные мероприятия, психологические разгрузки.

Режим труда и отдыха операторов должен организовываться в зависимости от вида и категории трудовой деятельности.

Категория А – работа по считыванию информации с дисплея с предварительным запросом.

Категория Б – работа по вводу информации.

Категория В – творческая работа в режиме диалога с ПЭВМ.

Продолжительность обеденного перерыва определяется Трудовым кодексом.[6]

Система физкультурно-оздоровительных мероприятий должна включать в себя различные формы производственной гимнастики: вводную гимнастику, физкультурную паузу, физкультурные минутки и другие малые формы (микрокомплексы, познотонические упражнения, профилактические приемы, самомассаж и др.), профилактическую и восстановительную гимнастику.

При этом следует знать, что каждое из указанных средств и форм воздействия на организм человека предназначено для определенной цели, а вся система в целом обеспечивает поддержание работоспособности на протяжении рабочего дня.

Помимо физкультурно-оздоровительных мероприятий для снижения нервно-эмоциональной нагрузки рекомендуется посещение комнат психологической разгрузки. Сеансы такой разгрузки в целях снятия усталости могут проводиться как во время регламентированных перерывов, так и по окончании рабочего дня. Они должны проводиться в специально оборудованном для этих целей помещении – комната психологической разгрузки.

Для защиты от статического электричества необходимо использовать нейтрализаторы и увлажнители. Полы должны иметь антистатическое покрытие.

Устройства визуального отображения генерируют несколько типов излучения, в том числе рентгеновское, радиочастотное, видимое и ультрафиолетовое. Однако уровни этих излучений достаточно низки и не превышают действующих норм.

Для снижения значений напряженности электрических и электростатических полей рекомендуется применение защитных фильтров, для снижения воздействия низкочастотных магнитных полей рекомендуется проводить полную внутреннюю защиту мониторов (со стороны, как экрана, так и корпуса), поскольку существующие приэкранные защитные фильтры в этом отношении являются неэффективными.

В помещениях необходимо контролировать уровень аэроионизации, ибо мягкое рентгеновское излучение, возникающее при напряжении на аноде 20-22 кВ, а также высокое напряжение на токоведущих участках схемы вызывают ионизацию воздуха с образованием положительных ионов, считающихся неблагоприятными для человека.

Оптимальным уровнем аэроионизации в зоне дыхания работающего считается содержание легких аэроионов обоих знаков от 1,5*102 до 5*101 в 1 см3 воздуха.

Для снижения влияния статического электричества рекомендуется устанавливать ионизаторами воздуха.

К непосредственной работе с ПЭВМ допускаются лица, не имеющие медицинских противопоказаний. Женщины со времени установления беременности и в период кормления грудью к выполнению всех видов работ, связанных с использованием ПЭВМ не допускаются.

Соблюдение вышеперечисленных норм и правил, внедрение новых технологий и научных разработок при организации рабочего места оператора ПЭВМ, позволит к минимуму снизить вредное воздействие производственных факторов на операторов и улучит условия рабочего места.

5.2 Расчет освещенности на рабочем месте

Рациональное освещение помещений - один из наиболее важных факторов, от которых зависит эффективность трудовой деятельности человека. Хорошее освещение необходимо для выполнения большинства

задач пользователя ЭВМ. Для того чтобы спланировать рациональную систему освещения, учитывается специфика рабочего задания, для которого создается система освещения, скорость и точность, с которой это рабочее задание должно выполняться, длительность его выполнения и различные изменения в условиях выполнения рабочих операций. [4]

Помещение, в котором находится рабочее место пользователя ЭВМ, имеет следующие характеристики:

- длина помещения 16 м;
- ширина помещения 6 м;
- высота 4 м;
- число окон 3;
- количество рабочих мест 3;
- окраска интерьера - белый потолок, бледно-зеленые стены;
- пол металлический, обтянутый линолеумом зеленого цвета.

В помещении, где находится рабочее место пользователя, используется смешанное освещение, т.е. сочетание естественного и искусственного освещения. В качестве естественного - боковое освещение через окна. Искусственное освещение используется при недостаточном естественном освещении. В данном помещении используется общее искусственное освещение. Расчет его осуществляется по методу светового потока с учетом потока, отраженного от стен и потолка.

Нормами для данных работ установлена необходимая освещенность рабочего места Ен = 300 [лк].

Общий световой поток определяется по формуле:

$$F_{общ} = \frac{Ен \times S \times z_1 \times z_2}{V \cdot N} \text{ [лм].}$$

где Ен - нормированная освещенность (Ен =300 [лк]);

S- площадь помещения;

z1 - коэффициент, учитывающий старение ламп и загрязнение светильников (z1=5).

z2 - коэффициент, учитывающий неравномерность освещения помещения (z2=1,1);

N – количество ламп в светильнике;

V- коэффициент использования светового потока, определяется в зависимости от коэффициентов отражения от стен, потолка, рабочих поверхностей, типов светильников и геометрии помещения.

Площадь помещения:

S=A*B=16*6=96 [м2].

Выберем коэффициент использования светового потока по следующим данным:

– коэффициент отражения побеленного потолка Rn = 70%;

– коэффициент отражения от стен, окрашенных в светлую краску Rcm=50%;

– коэффициент отражения от пола, покрытого линолеумом темного цвета Rp=10%;

– индекс помещения:

$$i = \frac{A*B}{h*(A+B)} = \frac{16*6}{4*(16*6)} = 1.1$$

Найденный коэффициент V= 0,34 (для светильника ПВЛ-1). Отсюда определяется общий световой поток:

$$F_{общ} = \frac{300*96*1{,}1*1{,}5}{0{,}34 \cdot 2} = 69882$$ [лм].

Для организации общего искусственного освещения выберем лампы типа ЛБ40. Люминесцентные лампы имеют ряд преимуществ перед лампами накаливания: их спектр ближе к естественному; они имеют большую экономичность (больше светоотдача) и срок службы (в 10-12 раз). Наряду с этим имеются и недостатки: их работа сопровождается иногда шумом, хуже работают при низких температурах, их нельзя применять во взрывоопасных

помещениях; имеют малую инерционность. Для нашего помещения люминесцентные лампы подходят. Световой поток одной лампы ЛБ40 составляет не менее Fn =3000 [лм]. Число ламп N, необходимых для организации общего освещения определяется по формуле:

$$N = \frac{F_{общ}}{F_{л}} = \frac{69882}{3000} = 24$$

В качестве светильников выбираем ПВЛ-1, 2х40 Вт. Таким образом, чтобы обеспечить световой поток Fобщ = 69882 [лм] надо использовать 12 светильников по 2 лампы ЛБ40 в каждом.

Рабочая высота подвеса определяется выражением:

$$h = H - h_1 = 4 - 0.7 - 0.1 = 3.2 \text{ [м]}.$$

Расстояние между центрами светильников:

$$L = 1{,}4 \cdot h = 1{,}4 \cdot 3.2 = 4{,}48 \text{ [м]}.$$

Число рядов светильников рассчитывается исходя из ширины помещения и расстояния между рядами:

$$n = \frac{B}{L} = \frac{6}{4.48} = 1.34 \approx 2$$

Размещаем светильники в два ряда.

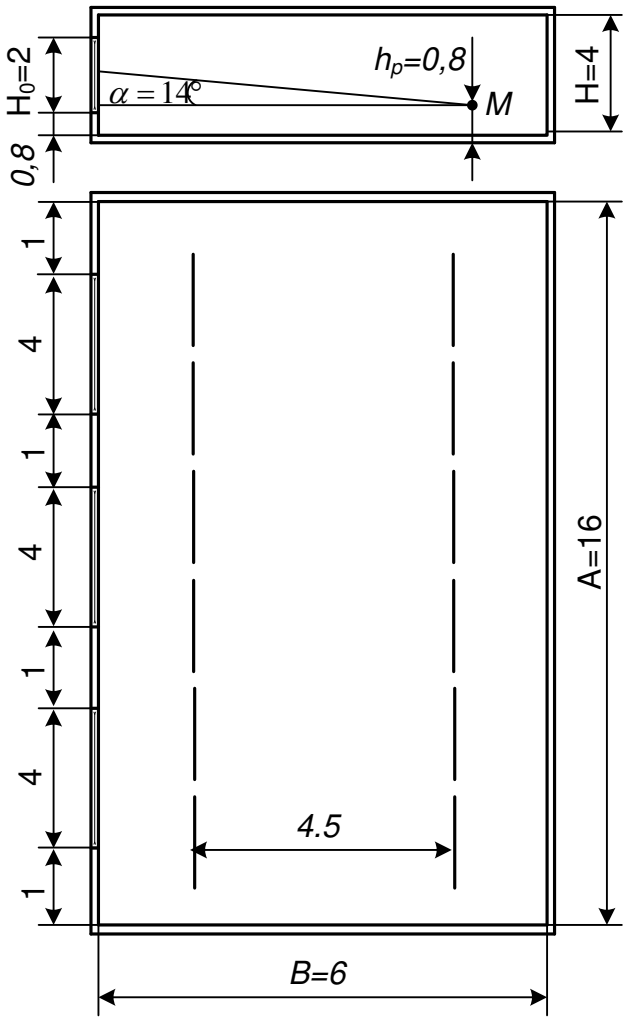


Рисунок 5.4 – Эскиз помещения с расположением светильников

5.3 Параметры микроклимата

Параметры микроклимата могут меняться в широких пределах, в то время как необходимым условием жизнедеятельности человека является поддержание постоянства температуры тела благодаря терморегуляции, т.е. способности организма регулировать отдачу тепла в окружающую среду. Принцип нормирования микроклимата – создание оптимальных условий для теплообмена тела человека с окружающей средой.

Вычислительная техника является источником существенных тепловыделений, что может привести к повышению температуры и снижению относительной влажности в помещении. В помещениях, где установлены компьютеры, должны соблюдаться определенные параметры микроклимата. В санитарных нормах установлены величины параметров микроклимата,

создающие комфортные условия. Эти нормы устанавливаются в зависимости от времени года, характера трудового процесса и характера производственного помещения (см. табл. 5.1).

Объем помещений, в которых размещены работники вычислительных центров, не должен быть меньше 19,5м3/человека с учетом максимального числа одновременно работающих в смену. Нормы подачи свежего воздуха в помещения, где расположены компьютеры, приведены в табл. 5.2.

Таблица 5.1 – Параметры микроклимата для помещений, где установлены компьютеры.

Период года	Параметр микроклимата	Величина
Холодный	Температура воздуха в помещении	22…24°С
	Относительная влажность	40…60%
	Скорость движения воздуха	до 0,1м/с
Теплый	Температура воздуха в помещении	23…25°С
	Относительная влажность	40…60%
	Скорость движения воздуха	0,1…0,2м/с

Таблица 5.2 – Нормы подачи свежего воздуха в помещения, где расположены компьютеры:

Характеристика помещения	Объемный расход подаваемого в помещение свежего воздуха, м3 /на одного человека в час
Объем до 20м3 на человека	Не менее 30

Продолжение таблицы 5.2

20…40м3 на человека	Не менее 20
Более 40м3 на человека	Естественная вентиляция

Для обеспечения комфортных условий используются как организационные методы (рациональная организация проведения работ в зависимости от времени года и суток, чередование труда и отдыха), так и технические средства (вентиляция, кондиционирование воздуха, отопительная система).

5.4 Шум и вибрация

Шум ухудшает условия труда оказывая вредное действие на организм человека. Работающие в условиях длительного шумового воздействия испытывают раздражительность, головные боли, головокружение, снижение памяти, повышенную утомляемость, понижение аппетита, боли в ушах и т.д. Такие нарушения в работе ряда органов и систем организма человека могут вызвать негативные изменения в эмоциональном состоянии человека вплоть до стрессовых. Под воздействием шума снижается концентрация внимания, нарушаются физиологические функции, появляется усталость в связи с повышенными энергетическими затратами и нервно-психическим напряжением, ухудшается речевая коммутация. Все это снижает работоспособность человека и его производительность, качество и безопасность труда. Длительное воздействие интенсивного шума (выше 80 дБ) на слух человека приводит к его частичной или полной потере. [3]

В табл. 5.3 указаны предельные уровни звука в зависимости от категории тяжести и напряженности труда, являющиеся безопасными в отношении сохранения здоровья и работоспособности.

Таблица 5.3 – Предельные уровни звука, дБ, на рабочих местах:

Категория напряженности труда	Категория тяжести труда		
Мало напряженный	Легкая	Средняя	Тяжелая
Умеренно напряженный	80	80	75
Напряженный	70	70	65
Очень напряженный	60	60	65
	50	50	55

5.5 Вывод

1. Выполнена организация рабочего места оператора ЭВМ с учетом требований безопасности;

2. Из произведенного в данном разделе расчета следует, что для нормальной работы пользователя рабочего места с видеотерминальным устройством необходимо общее освещение помещения со световым потоком 69882 [лм], для чего необходимо наличие 12 светильников типа ПВЛ-1 с 2-мя лампами типа ЛБ40 каждый. Светильники должны быть расположены в два ряда, что является наиболее оптимальным решением.

3. Соблюдение условий, определяющих оптимальную организацию рабочего места инженера - программиста, позволит сохранить хорошую работоспособность в течение всего рабочего дня. Повысит, как в количественном, так и в качественном отношениях производительность труда программиста, что в свою очередь будет способствовать быстрейшей разработке и отладке программного продукта.

Заключение

В ходе выполнения работы по разработке Web-приложения «Планировщик задач для инклюзивного образования» было сделано:

1. Анализ предметной области;
2. Рассмотрены технологии систем различных производителей;
3. Рассмотрены существующие реализации подобных систем и был сделан анализ, в котором были выявлены достоинства и недостатки;
4. Разработан алгоритм работы и был сделан выбор программного обеспечения для разработки приложения;.
5. Разработана база данных, позволяющая хранить в себе массивы данных;
6. Разработано программное обеспечение «Планировщик задач для инклюзивного образования».

Применение Web-приложения упростит регулирование данных в расписании учителей, сократит временные затраты на его составление и просмотр данных учениками.

Использование в программном комплексе СУБД, позволяет обеспечить хранение вводимых данных.

.

Список использованных источников

1 Интеллектуальные модели анализа информации: курс лекций 2017 BaseGroup Labs.

2 Елманова Н. Введение в технологии PHP. / Н. Елманова, Федоров А. – М.: Диалог-МИФИ, 2017.

3 П. Блаттнер - Special Edition Using MySQL. – М.: Вильямс, 2016.

4 Романов Станислав. Обзор Remember the Milk. http://gadgetick.com/newsnew-4176.html , 2016.

5 Мацаяшек А. Анализ требований и проектирование системы. Разработка информационных систем с использованием UML / А. Мацаяшек – М.: Издательский дом «Вильямс», 2015.

6 Фаулер М. UML. Основы / М. Фаулер, К. Скотт – СПб: Символ-Плюс, 2014.

7 Дейтел Х. – PHP in DB / Х. Дейтел, Дейтел П., Листфид Дж., Нието Т., Йегер Ш., Златкина М. - СПб: БХВ-Петербург, 2016.

8 Фримен А. –Язык интегрированных запросов в PHP 2016 для профессионалов / А. Фримен, Раттц-мл Д. – М.: Вильямс, 2018.

9 Фримен А. –MVC Framework с примерами на PHP для профессионалов / Сандерсон С. - М.: Вильямс, 2017.

10 Флэнаган Д. – JavaScript Подробное руководство – СПб: Символ-Плюс, 2018.

11 Мандел Т. Разработка пользовательского интерфейса / Т. Мандел – М.: ДМК Пресс, 2018.

12 Гома Х. UML. Проектирование систем реального времени параллельных и распределенных приложений / Х. Гома – М.: ДМК Пресс, 2015.

13 Котляров В. П. Основы тестирования программного обеспечения: Учебное пособие / В. П. Котляров, Т. В. Коликова – М.: Бином. Лаборатория знаний, 2016.

14 Макгрегор Дж. Тестирование объектно-ориентированного программного обеспечения / Дж. Макгрегор, Д. Сайкс. – К.: ООО «ТИД «ДС», 2015.

15 Дзарасов М.Э. Ответственность по нормам трудового права. Учебное пособие, 2018.

16 «Санитарно-эпидемиологические требования к физическим факторам на рабочих местах». СанПиН 2.2.4.3359-16.

17 Бурлак Г.Г. «Безопасность работы на компьютере; организация труда на предприятиях информационного обслуживания», 2012.

18 «Естественно и искусственное освещение». СНиП 23-05-2010.

19 Н.И. Бобков, Голованова Т.В. «Охрана труда на ВЦ», МАИ, 1998.

20 Г.Ф. Денисенко «Охрана труда», высшая школа 2015.

КОД ПРОГРАММЫ (СЕРВЕРНАЯ ЧАСТЬ)

Приложение А

(обязательное)

Файл .htaccess

```
Options +FollowSymLinks

RewriteEngine on
# if a directory or a file exists, use it directly
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
# otherwise forward it to index.php
RewriteRule . index.html
```

Файл DB.php

```
<?php

class db{
   protected $host = "localhost";
   protected $dbname = "stock";
   protected $user = "root";
   protected $pass = "";
   protected $dbh;

   function __construct() {

      try {

         $this->dbh = new PDO("mysql:host=$this->host;dbname=$this->dbname", $this->user,
$this->pass, array(PDO::MYSQL_ATTR_INIT_COMMAND => "SET NAMES 'utf8'"));
      }
      catch (PDOException $e) {

         echo $e->getMessage();
      }
   }

   public function closeConnection() {

      $this->dbh = null;
   }
}
```

Файл Model.php

```
<?php
include_once "db.php";
include_once "myphp-shedule.php";
session_start();

class Model extends DB {
    /*
     * Функция авторизации
     */
    public function setLogin($data) {
        $login = $data["username"];
        $pass = $data["password"];
        //$pass1 = md5($pass);
        $STH = $this->dbh->prepare("select * from users
        left join `group` on `group`.id_group = users.id_group
        where (users.nickName = '$login') and (users.password = '$pass')");
        $STH->execute();
        $_group = $STH ->fetchColumn(6);
        $STH->execute();
        if($STH->rowCount() > 0){
            $results = $STH->fetchAll(PDO::FETCH_ASSOC);
            $_SESSION = $results[0];
            $_VIS=$this->dbh->prepare("select id from routes where id_group = '$_group' " );
            $_VIS->execute();
            if($_VIS->rowCount() > 0) {
                $_visit_res = $_VIS ->fetchAll(PDO::FETCH_ASSOC);
                $_SESSION['visit'] = array_map(function($o) { return $o["id"]; }, $_visit_res);
            }
            else {
                $_SESSION['visit'] = [];
            }
            $resault = (object) array("success" => true, "message" => "Ok", "group" => $_group) ;
            return ($resault);
        }
        $res = false;

        return ($res);
    }
    /*
     * Функция выхода
     */
    public function getLogout(){
        session_destroy();
        $res = (object) array("res" => "true");
        return ($res);
    }

        public function getRoutes(){

                $group = $_SESSION['id_group'];

                $_VIS=$this->dbh->prepare("select * from routes where id_group = '$group' " );
```

```
                $_VIS->execute();
                if($_VIS->rowCount() > 0) {
                        $_visit_res = $_VIS ->fetchAll(PDO::FETCH_ASSOC);

                }
                else {
                        $_visit_res = [];
                }

        return ($_visit_res);
           }

    public function setShedule() {

        $shedule = new Shedule_Database();
        $date = date ('Y-m-d H:i');
        $res = $shedule->sheduleDir . '/' . $shedule->sheduleFile .'.gz';
        $file =$shedule->sheduleFile .'.gz';
        $dir = $shedule->sheduleDir;
        $STH=$this->dbh->prepare("insert into shedules(name, createDate, path) VALUES
('$file','$date','$dir')");
        $STH->execute();
        return($res);

    }

          public function getShedules() {
                $STH = $this->dbh->prepare("select * from shedules");
                $STH->execute();
                $results["data"] = $STH->fetchAll(PDO::FETCH_ASSOC);
                return ($results);
    }

          public function deleteShedule($id) {

                $STH = $this->dbh->prepare("DELETE FROM shedules WHERE id= '$id'");
        $STH->execute();
        $results = $STH->fetchAll(PDO::FETCH_ASSOC);

        return ("Shedule deleted");
           }

    /*
     * Функция получения списка пользователей
     */
    public function getUsers(){
        $STH = $this->dbh->prepare("select users.id_user as id, users.nickName, users.f_name,
users.password, users.l_name, users.s_name, group.Name as 'group' from users
                 join `group` on `group`.id_group = users.id_group");
        $STH->execute();
        $results["data"] = $STH->fetchAll(PDO::FETCH_ASSOC);
        return ($results);
```

```
    }

    public function getUser(){
                $temp =
'{"success":true,"user":{"id":null,"username":"","f_name":"","s_name":"","l_name":"","permissi
ons":{"visit":[],"role":""},"avatar":""}}';
                $results = json_decode($temp);

                $results->user->id = $_SESSION["id_user"];
                $results->user->username = $_SESSION["nickName"];
                $results->user->f_name = $_SESSION["f_name"];
                $results->user->s_name = $_SESSION["s_name"];
                $results->user->l_name = $_SESSION["l_name"];
                $results->user->permissions->visit = $_SESSION["visit"];
                $results->user->permissions->role = $_SESSION["Name"];
        return ($results);
    }

        /**/

    public function setUser($data) {
                $login = $data["nickName"];
        $password = $data["password"];
                $fname = $data["name"];
                $lname = $data["f_name"];
                $sname = $data["l_name"];
                $id_group = $data["group"];

        $STH = $this->dbh->prepare("insert into
users(nickName,password,f_name,l_name,s_name,id_group)
VALUES('$login','$password','$fname','$lname','$sname',$id_group)");
        $STH->execute();
        $results = $STH->fetchAll(PDO::FETCH_ASSOC);
        $res = (object) array("success" => true, "message" => "Ok");
        return ($res);
    }
    public function changeUser($id, $data){
                $login = $data["nickName"];
        $password = $data["password"];
                $fname = $data["name"];
                $lname = $data["f_name"];
                $sname = $data["l_name"];
                $id_group = $data["group"];
        $STH = $this->dbh->prepare("update users SET nickName ='$login', password
='$password',f_name = '$fname',l_name = '$lname',s_name = '$sname',id_group = '$id_group'
where id_user = '$id'");
        $STH->execute();
        $results = $STH->fetchAll(PDO::FETCH_ASSOC);

        return ("User changed");
    }
    public function deleteUser($id) {
```

```
        $STH = $this->dbh->prepare("DELETE FROM users WHERE id_user= '$id'");
        $STH->execute();
        $results = $STH->fetchAll(PDO::FETCH_ASSOC);
        return ("User deleted");
    }
    public function getCategories(){
        $STH = $this->dbh->prepare("select * FROM category");
        $STH->execute();
        $results = $STH->fetchAll(PDO::FETCH_ASSOC);
        return ($results);
    }
    public function getCategory($id){
        $STH = $this->dbh->prepare("select * FROM category where id_category ='$id'");
        $STH->execute();
        $results = $STH->fetchAll(PDO::FETCH_ASSOC);
        return ($results);
    }
    public function setCategory($name){
        $STH = $this->dbh->prepare("insert into category(Name) VALUES($name)");
        $STH->execute();
        $results = $STH->fetchAll(PDO::FETCH_ASSOC);
        return ($results);
    }
    public function changeCategory($id,$Name){
        $STH = $this->dbh->prepare("update category SET Name ='$Name' where id_category =
'$id'");
        $STH->execute();
        $results = $STH->fetchAll(PDO::FETCH_ASSOC);
        return ($results);
    }
    public function deleteCategory($id){
        $STH = $this->dbh->prepare("DELETE FROM category WHERE id_category = '$id'");
        $STH->execute();
        $results = $STH->fetchAll(PDO::FETCH_ASSOC);
        return ($results);
    }
    public  function  getEvents(){
        $STH = $this->dbh->prepare("select * FROM event");
        $STH->execute();
        $results["data"] = $STH->fetchAll(PDO::FETCH_ASSOC);
        return ($results);
    }
    public  function  getEvent($id){
        $STH = $this->dbh->prepare("select * FROM event where id_event = '$id'");
        $STH->execute();
        $results = $STH->fetchAll(PDO::FETCH_ASSOC);
        return ($results);
    }
    public  function  getEventsByCategory($id){
        $STH = $this->dbh->prepare("select Name FROM event where id_category = '$id'");
        $STH->execute();
```

```
        $results = $STH->fetchAll(PDO::FETCH_ASSOC);
        return ($results);
    }
    public function
setEvent($name,$id_category,$id_measure,$price,$count,$id_brand,$id_producer){
        $STH = $this->dbh->prepare("insert into
event(Name,id_category,id_measure,price,count,id_brand,id_producer)
VALUES($name,$id_category,$id_measure,$price,$count,$id_brand,$id_producer)");
        $STH->execute();
        $results = $STH->fetchAll(PDO::FETCH_ASSOC);
        return ($results);
    }
    public function
changeEvent($id,$name,$id_category,$id_measure,$price,$count,$id_brand,$id_producer){
        $STH = $this->dbh->prepare("update event SET Name ='$name', id_category
='$id_category', id_measure = '$id_measure', price = '$price', count = '$count', id_brand =
'$id_brand', id_producer = '$id_producer'  where id_event = '$id'");
        $STH->execute();
        $results = $STH->fetchAll(PDO::FETCH_ASSOC);
        return ($results);
    }
    public function deleteEvent($id){
        $STH = $this->dbh->prepare("DELETE FROM event WHERE id_event = '$id'");
        $STH->execute();
        $results = $STH->fetchAll(PDO::FETCH_ASSOC);
        return ($results);
    }
/*
* Функция получения цены товара
*/
    public function getPrice($id_event) {
        $STH = $this->dbh->prepare("select price from event where id_event ='$id_event'");
        $STH->execute();
        $results = $STH->fetchAll(PDO::FETCH_ASSOC);
        return ($results);
    }
    /*
     * Функция получения списка издетелей
     */
    public function getProducers() {
        $STH = $this->dbh->prepare("select * from producer");
        $STH->execute();
        $results = $STH->fetchAll(PDO::FETCH_ASSOC);
        return ($results);
    }
    public function getProducer($id) {
        $STH = $this->dbh->prepare("select * from producer where id_producer ='$id'");
        $STH->execute();
        $results = $STH->fetchAll(PDO::FETCH_ASSOC);
        return ($results);
    }
    public function setProducer($Name,$Phone,$Address) {
```

```
        $STH = $this->dbh->prepare("insert into producer(Name,Phone,Address)
VALUES('$Name','$Phone',$Address)");
        $STH->execute();
        $results = $STH->fetchAll(PDO::FETCH_ASSOC);
        return ($results);
    }
    public function deleteProducer($id) { //delete дописать
        $STH = $this->dbh->prepare("DELETE FROM producer WHERE id_producer = '$id'");
        $STH->execute();
        $results = $STH->fetchAll(PDO::FETCH_ASSOC);
        return ($results);
    }
    public function changeProducer($id,$name,$phone,$address) { //delete дописать
        $STH = $this->dbh->prepare("update event SET Name ='$name', Phone ='$phone', Address
= '$address' where id_producer = '$id'");
        $STH->execute();
        $results = $STH->fetchAll(PDO::FETCH_ASSOC);
        return ($results);
    }

/*
 * Функция получения списка брендов
 */
    public function getBrands() {
    $STH = $this->dbh->prepare("select * from brand ");
    $STH->execute();
    $results = $STH->fetchAll(PDO::FETCH_ASSOC);
    return ($results);
    }
    /*
     * Функция добавления бенда
     */
    public function setBrand($id_producer,$name) {
        $STH = $this->dbh->prepare("insert into brand(Name,id_producer)
VALUES($name,$id_producer)");
        $STH->execute();
        $results = $STH->fetchAll(PDO::FETCH_ASSOC);
        return ($results);
    }
    public function changeBrand($id,$id_producer,$name) {
        $STH = $this->dbh->prepare("update brand SET Name ='$name', id_producer
='$id_producer' where id_brand = '$id'");
        $STH->execute();
        $results = $STH->fetchAll(PDO::FETCH_ASSOC);
        return ($results);
    }
    public function deleteBrand($id) {
        $STH = $this->dbh->prepare("DELETE FROM brand WHERE id_brand = '$id'");
        $STH->execute();
        $results = $STH->fetchAll(PDO::FETCH_ASSOC);
        return ($results);
    }
```

```
    public function getBrandByProducer($producer) {
        $STH = $this->dbh->prepare("select * from brand where id_producer = '$producer'");
        $STH->execute();
        $results = $STH->fetchAll(PDO::FETCH_ASSOC);
        return ($results);
    }
    /*
     * Функция установки статуса
    */
    public function setStatus($id_order,$id_status) {
        $date = date('Y-m-d H:i:s');
        $STH = $this->dbh->prepare("insert into order_logs(id_order,id_producer,date)
VALUES('$id_order','$id_status',$date)");
        $STH->execute();
        $results = $STH->fetchAll(PDO::FETCH_ASSOC);
        return ($results);
    }
   /*
 * Функция установки привилегии
*/
    public function setPrivilege($id_user,$id_category) {
        $STH = $this->dbh->prepare("insert into privileges(id_category,id_user) VALUES
($id_category,$id_user)");
        $STH->execute();
        $results = $STH->fetchAll(PDO::FETCH_ASSOC);
        return ($results);
    }
    public function deletePrivilegeUser($id_user) {
        $STH = $this->dbh->prepare("DELETE FROM priveleges WHERE id_user = '$id_user'");
        $STH->execute();
        $results = $STH->fetchAll(PDO::FETCH_ASSOC);
        return ($results);
    }

    public function pathcPrivilege($id_user,$id_category,$id) {
        $STH = $this->dbh->prepare("update priveleges SET Name ='$id_user', id_category
='$id_category' where id_privilege = '$id'");
        $STH->execute();
        $results = $STH->fetchAll(PDO::FETCH_ASSOC);
        return ($results);
    }
    public function getPrivilegeByUser($id_user) {
        $STH = $this->dbh->prepare("select * from priveleges where id_user = '$id_user'");
        $STH->execute();
        $results = $STH->fetchAll(PDO::FETCH_ASSOC);
        return ($results);
    }
    public function getPrivileges() {
        $STH = $this->dbh->prepare("select * from priveleges");
        $STH->execute();
        $results = $STH->fetchAll(PDO::FETCH_ASSOC);
        return ($results);
```

```
    }
    public function getCategoriesByUser() {
        $id=$_SESSION["id_user"];
        $STH = $this->dbh->prepare("select category.`Name` from `privileges` left join category
on category.id_category = `privileges`.id_category where `privileges`.id_user = '$id'");
        $STH->execute();
        $results = $STH->fetchAll(PDO::FETCH_ASSOC);
        return ($results);
    }
    public function setEventInCart($data){

                $id_event = $data["event_id"];
                $_count = $data["count"];

                $id_user = $_SESSION['id_user'];

        $STH = $this->dbh->prepare("insert into cart(id_event,count,id_user)
VALUES($id_event,$_count,$id_user)");
        $STH->execute();
        $results = $STH->fetchAll(PDO::FETCH_ASSOC);
        $res = (object) array("success" => true, "message" => "Ok");
        return ($res);
    }
    public function getUserCart() {///по отдельному заказу?
        $user = $_SESSION['id_user'];
        $STH = $this->dbh->prepare("select event.Name as `title`, cart.id_event, sum(cart.count) as
`count` from `cart` join `event` on `event`.`id_event` = `cart`.`id_event` where id_user = '$user'
GROUP BY cart.id_event");
        $STH->execute();
        $results["data"] = $STH->fetchAll(PDO::FETCH_ASSOC);
        return ($results);
    }
    public function patchEventInCart($id_cart,$id_event,$count) {
        $STH = $this->dbh->prepare("update cart SET count = '$count' where id_cart = '$id_cart'
and id_event ='$id_event'");
        $STH->execute();
        $results = $STH->fetchAll(PDO::FETCH_ASSOC);
        return ($results);
    }
    public function getClientHistoryOrder($id) {
        $STH = $this->dbh->prepare("select id_order from order where id_client = '$id'");
        $STH->execute();
        $results = $STH->fetchAll(PDO::FETCH_ASSOC);
        return ($results);
    }
    public function getUserHistoryOrder() {
        $id_user=$_SESSION['id_user'];
        $STH = $this->dbh->prepare("select `order`.id_order as id, `order_logs`.id_status as status,
`order_logs`.date from `order`
                join `order_logs` on `order_logs`.id_order = `order`.id_order
                where `order`.id_user = '$id_user'");
        $STH->execute();
```

```
        $results["data"] = $STH->fetchAll(PDO::FETCH_ASSOC);

        return ($results);
    }
    public function patchClientByCart($id_cart,$address,$phone,$fname,$sname,$lname,$email)
{
        $STH = $this->dbh->prepare("update client left join `order` on `order`.id_client =
client.id_client  SET address = '$address', phone = '$phone', f_name = '$fname', s_name =
'$sname', l_name ='$lname',email ='$email'   where `order`.id_cart = '$id_cart' ");
        $STH->execute();
        $results = $STH->fetchAll(PDO::FETCH_ASSOC);
        return ($results);
    }


}
```

Файл Request.php

```
<?php
error_reporting(E_ERROR | E_WARNING | E_PARSE);

include_once("db.php"); //Подключаем класс для работы с бд
include_once("model.php"); //Подключаем класс с функционалом нашего приложения

header('Access-Control-Allow-Origin: *');
if (isset($_SERVER['HTTP_ORIGIN'])) {
        header("Access-Control-Allow-Origin: {$_SERVER['HTTP_ORIGIN']}");
        header('Access-Control-Allow-Credentials: true');
        header('Access-Control-Max-Age: 86400');
}

if ($_SERVER['REQUEST_METHOD'] == 'OPTIONS') {
if (isset($_SERVER['HTTP_ACCESS_CONTROL_REQUEST_METHOD']))
header("Access-Control-Allow-Methods: GET, POST, PUT, DELETE, OPTIONS");
if (isset($_SERVER['HTTP_ACCESS_CONTROL_REQUEST_HEADERS']))
header("Access-Control-Allow-Headers:
{$_SERVER['HTTP_ACCESS_CONTROL_REQUEST_HEADERS']}");
}

$con    = new DB();
$model  = new Model();
$postData = file_get_contents('php://input'); //Получаем данные, которые отправили на
сервер
$data = json_decode($postData, true); //Декодируем из json в массив
$url = $_SERVER['REQUEST_URI'];
$parts = explode('/', $url);
$method = $_SERVER['REQUEST_METHOD'];

$mode = $parts[2];
$params = isset($parts[3]) ? $parts[3] : '';
```

```
$api =  ucwords($mode);

function response($d) {
        if($d){
                exit(json_encode($d));
        }
        else {
                header('HTTP/1.0 400 Bad Request', true);
                exit();
        }
}

switch ($method)
{
   case 'GET':
                $grub = 'get'.$api;
                $drar = $model->$grub($params);
                response($drar);
                break;
   case 'POST':
      $grub = 'set'.$api;
                $drar = $model->$grub($data);
                response($drar);
                break;
         case 'DELETE':
      $grub = 'delete'.$api;
                $drar = $model->$grub($params);
                response($drar);
                break;
         case 'PATCH':
      $grub = 'change'.$api;
                $drar = $model->$grub($params, $data);
                response($drar);
                break;
         default:
                break;
}


/*
 * Срабатывает при загрузке файла
 */
if(isset($_FILES['file']))
{
   $uploaddir = '../img/';

   if(!is_dir($uploaddir)) {
      mkdir($uploaddir);
   }
   $uploadfile = $uploaddir . iconv("utf-8","windows-1251",$_FILES['file']['name']);
```

```
    if (move_uploaded_file($_FILES['file']['tmp_name'], $uploadfile)) {
      $res =  (object) array("res" => "true");
       exit(json_encode($res));
    } else {
      $res =  (object) array("res" => "false");
       exit(json_encode($res));
    }
}
```

КОД ПРОГРАММЫ (КЛИЕНТСКАЯ ЧАСТЬ)

Приложение Б

(обязательное)

Глобальное пространство
Файл view.js

```
import React, { PureComponent, Fragment } from 'react'
import PropTypes from 'prop-types'
import { connect } from 'dva'
import { Helmet } from 'react-helmet'
import { Loader } from 'components'
import { queryLayout } from 'utils'
import NProgress from 'nprogress'
import config from 'utils/config'
import withRouter from 'umi/withRouter'

import PublicLayout from './PublicLayout'
import PrimaryLayout from './PrimaryLayout'
import './BaseLayout.less'

const LayoutMap = {
primary: PrimaryLayout,
public: PublicLayout,
}

@withRouter
@connect(({ loading }) => ({ loading }))
class BaseLayout extends PureComponent {
previousPath = ''

render() {
const { loading, children, location } = this.props
const Container = LayoutMap[queryLayout(config.layouts, location.pathname)]

const currentPath = location.pathname + location.search
if (currentPath !== this.previousPath) {
NProgress.start()
}

if (!loading.global) {
NProgress.done()
this.previousPath = currentPath
}

return (
<Fragment>
```

```
<Helmet>
<title>{config.siteName}</title>
</Helmet>
<Loader fullScreen spinning={loading.effects['app/query']} />
<Container>{children}</Container>
</Fragment>
)
}
}

BaseLayout.propTypes = {
loading: PropTypes.object,
}

export default BaseLayout

/* global document */
import React, { PureComponent, Fragment } from 'react'
import PropTypes from 'prop-types'
import withRouter from 'umi/withRouter'
import { connect } from 'dva'
import { MyLayout } from 'components'
import { BackTop, Layout, Drawer } from 'antd'
import { GlobalFooter } from 'ant-design-pro'
import { enquireScreen, unenquireScreen } from 'enquire-js'
import { config, pathMatchRegexp, langFromPath } from 'utils'
import Error from '../pages/404'
import styles from './PrimaryLayout.less'

const { Content } = Layout
const { Header, Bread, Sider } = MyLayout

@withRouter
@connect(({ app, loading }) => ({ app, loading }))
class PrimaryLayout extends PureComponent {
state = {
isMobile: false,
}

componentDidMount() {
this.enquireHandler = enquireScreen(mobile => {
const { isMobile } = this.state
if (isMobile !== mobile) {
this.setState({
isMobile: mobile,
})
}
})
}

componentWillUnmount() {
unenquireScreen(this.enquireHandler)
```

```
}

render() {
const { app, location, dispatch, children } = this.props
const {
user,
theme,
routeList,
collapsed,
permissions,
notifications,
} = app
const { isMobile } = this.state
const { onCollapseChange } = this

const lang = langFromPath(location.pathname)
const newRouteList =
lang !== 'en'
? routeList.map(item => {
const { name, ...other } = item
return {
...other,
name: (item[lang] || {}).name || name,
}
})
: routeList

const currentRoute = newRouteList.find(
_ => _.route && pathMatchRegexp(_.route, location.pathname)
)

const hasPermission = currentRoute
? permissions.visit.includes(currentRoute.id)
: false

const menus = newRouteList.filter(_ => _.menuParentId !== '-1')


}

if(permissions) {
headerProps = {...headerProps, notifications}
}

return (
<Fragment>
<Layout>
{isMobile ? (
<Drawer
maskClosable
closable={false}
onClose={onCollapseChange.bind(this, !collapsed)}
```

```
visible={!collapsed}
placement="left"
width={200}
style={{
padding: 0,
height: '100vh',
}}
>
<Sider {...siderProps} collapsed={false} />
</Drawer>
) : (
<Sider {...siderProps} />
)}
<div
className={styles.container}
style={{ paddingTop: config.fixedHeader ? 72 : 0 }}
id="primaryLayout"
>
<Header {...headerProps} />
<Content className={styles.content}>
<Bread routeList={newRouteList} />
{hasPermission ? children : <Error />}
</Content>
<BackTop
className={styles.backTop}
target={() => document.querySelector('#primaryLayout')}
/>
<GlobalFooter
className={styles.footer}
copyright={config.copyright}
/>
</div>
</Layout>
</Fragment>
)
}
}

PrimaryLayout.propTypes = {
children: PropTypes.element.isRequired,
location: PropTypes.object,
dispatch: PropTypes.func,
app: PropTypes.object,
loading: PropTypes.object,
}

export default PrimaryLayout

import React, { Component } from 'react'
import withRouter from 'umi/withRouter'
import { LocaleProvider } from 'antd'
import { I18nProvider } from '@lingui/react'
```

```
import { langFromPath, defaultLanguage } from 'utils'
import zh_CN from 'antd/lib/locale-provider/zh_CN'
import en_US from 'antd/lib/locale-provider/en_US'

import BaseLayout from './BaseLayout'

const languages = {
zh: zh_CN,
en: en_US,
}

@withRouter
class Layout extends Component {
state = {
catalogs: {},
}

language = defaultLanguage

componentDidMount() {
const language = langFromPath(this.props.location.pathname)
this.language = language
this.loadCatalog(language)
}

shouldComponentUpdate(nextProps, nextState) {
const language = langFromPath(nextProps.location.pathname)
const preLanguage = this.language
const { catalogs } = nextState

if (preLanguage !== language && !catalogs[language]) {
this.loadCatalog(language)
this.language = language
return false
}
this.language = language

return true
}

loadCatalog = async language => {
const catalog = await import(/* webpackMode: "lazy", webpackChunkName: "i18n-[index]" */
`@lingui/loader!../locales/${language}/messages.json`)

this.setState(state => ({
catalogs: {
...state.catalogs,
[language]: catalog,
},
}))
}
```

```
render() {
const { location, children } = this.props
const { catalogs } = this.state

let language = langFromPath(location.pathname)

if (!catalogs[language]) language = defaultLanguage

return (
<LocaleProvider locale={languages[language]}>
<I18nProvider language={language} catalogs={catalogs}>
<BaseLayout>{children}</BaseLayout>
</I18nProvider>
</LocaleProvider>
)
}
}

export default Layout
```

Файл action.js

```
onCollapseChange = collapsed => {
this.props.dispatch({
type: 'app/handleCollapseChange',
payload: collapsed,
})
}

let headerProps = {
menus,
collapsed,
onCollapseChange,
avatar: user.avatar,
name: user.f_name,
fixed: config.fixedHeader,
onAllNotificationsRead() {
dispatch({ type: 'app/allNotificationsRead' })
},
onSignOut() {
dispatch({ type: 'app/signOut' })
},

const siderProps = {
theme,
menus,
isMobile,
collapsed,
onCollapseChange,
```

```
onThemeChange(theme) {
dispatch({
type: 'app/handleThemeChange',
payload: theme,
})
},
}
```

Файл model.js

```
/* global window */

import { router } from 'utils'
import { stringify } from 'qs'
import store from 'store'
import { ROLE_TYPE } from 'utils/constant'
import { queryLayout, pathMatchRegexp } from 'utils'
import { CANCEL_REQUEST_MESSAGE } from 'utils/constant'
import api from 'api'
import config from 'config'

const { queryRouteList, logoutUser, queryUserInfo } = api

export default {
namespace: 'app',
state: {
user: {},
permissions: {
visit: [],
},
routeList: [
{
id: '1',
icon: 'laptop',
name: 'Dashboard',
router: '/dashboard',
},
],
locationPathname: '',
locationQuery: {},
theme: store.get('theme') || 'light',
collapsed: store.get('collapsed') || false,
notifications: [
{
title: 'Зарегистрирован новый пользователь.',
date: new Date(Date.now() - 10000000),
},
{
```

```
title: 'Application has been approved.',
date: new Date(Date.now() - 50000000),
},
],
},
subscriptions: {
setupHistory({ dispatch, history }) {
history.listen(location => {
dispatch({
type: 'updateState',
payload: {
locationPathname: location.pathname,
locationQuery: location.query,
},
})
})
},

setupRequestCancel({ history }) {
history.listen(() => {
const { cancelRequest = new Map() } = window

cancelRequest.forEach((value, key) => {
if (value.pathname !== window.location.pathname) {
value.cancel(CANCEL_REQUEST_MESSAGE)
cancelRequest.delete(key)
}
})
})
},

setup({ dispatch }) {
dispatch({ type: 'query' })
},
},
effects: {
*query({ payload }, { call, put, select }) {
const { success, user } = yield call(queryUserInfo, payload)
const { locationPathname } = yield select(_ => _.app)

if (success && user) {
const { list } = yield call(queryRouteList)
const { permissions } = user
let routeList = list

routeList = list.filter(item => {
const cases = [
permissions.visit.includes(item.id),
item.mpid
? permissions.visit.includes(item.mpid) || item.mpid === '-1'
: true,
item.bpid ? permissions.visit.includes(item.bpid) : true,
```

```
]
return cases.every(_ => _)
})

yield put({
type: 'updateState',
payload: {
user,
permissions,
routeList,
},
})
if (pathMatchRegexp(['/','/login'], window.location.pathname)) {

if(permissions.role === 'Admin') {
router.push({
pathname: '/user',
})
}
else {
router.push({
pathname: '/dashboard',
})
}
}
} else if (queryLayout(config.layouts, locationPathname) !== 'public') {
router.push({
pathname: '/login',
search: stringify({
from: locationPathname,
}),
})
}
},

*signOut({ payload }, { call, put }) {
const data = yield call(logoutUser)
if (data.success) {
yield put({
type: 'updateState',
payload: {
user: {},
permissions: { visit: [] },
menu: [
{
id: '1',
icon: 'laptop',
name: 'Dashboard',
router: '/dashboard',
},
],
},
```

```
})
yield put({ type: 'query' })
} else {
throw data
}
},
},
reducers: {
updateState(state, { payload }) {
return {
...state,
...payload,
}
},

handleThemeChange(state, { payload }) {
store.set('theme', payload)
state.theme = payload
},

handleCollapseChange(state, { payload }) {
store.set('collapsed', payload)
state.collapsed = payload
},

allNotificationsRead(state) {
state.notifications = []
},
},
}
```

Файл service_routes.js

```
export default {
queryRouteList: '/routes',

queryUserInfo: '/user',
logoutUser: '/logout',
loginUser: 'POST /login',

queryUser: '/user/:id',
queryUserList: '/users',
updateUser: 'Patch /user/:id',
createUser: 'POST /user',
removeUser: 'DELETE /user/:id',
removeUserList: 'POST /users/delete',

queryEventList: '/events',
createEvent: 'POST /eventInCart',

queryCart: '/userCart',
```

```
queryDashboard: '/userHistoryOrder',

queryShedules: '/shedules',
createShedule: 'POST /shedule',
removeShedule: 'DELETE /shedule/:id',
removeShedulesList: 'POST /shedules/delete',

}
```

Файл login.js (View)

```
import React, { PureComponent, Fragment } from 'react'
import PropTypes from 'prop-types'
import { connect } from 'dva'
import { Button, Row, Form, Input } from 'antd'

import { Trans, withI18n } from '@lingui/react'

import config from 'utils/config'

import styles from './index.less'
const FormItem = Form.Item

@withI18n()
@connect(({ loading }) => ({ loading }))
@Form.create()
class Login extends PureComponent {

render() {
const { loading, form, i18n } = this.props
const { getFieldDecorator } = form

return (
<Fragment>
<div className={styles.form}>
<div className={styles.logo}>
<span>{config.siteName}</span>
</div>
<form>
<FormItem hasFeedback>
{getFieldDecorator('username', {
rules: [
{
required: true,
},
],
```

```
})(
<Input
onPressEnter={this.handleOk}
placeholder={i18n.t`Username`}
/>
)}
</FormItem>
<FormItem hasFeedback>
{getFieldDecorator('password', {
rules: [
{
required: true,
},
],
})(
<Input
type="password"
onPressEnter={this.handleOk}
placeholder={i18n.t`Password`}
/>
)}
</FormItem>
<Row>
<Button
type="primary"
onClick={this.handleOk}
loading={loading.effects.login}
>
<Trans>Sign in</Trans>
</Button>
</Row>
</form>
</div>
</Fragment>
)
}
}

Login.propTypes = {
form: PropTypes.object,
dispatch: PropTypes.func,
loading: PropTypes.object,
}

export default Login
```

Файл clogin.js (Controller)

```
handleOk = () => {
const { dispatch, form } = this.props
const { validateFieldsAndScroll } = form
validateFieldsAndScroll((errors, values) => {
```

```
if (errors) {
return
}
dispatch({ type: 'login/login', payload: values })
})
}
```

Файл mlogin.js (Model)

```
import { router, pathMatchRegexp } from 'utils'
import api from 'api'

const { loginUser } = api

export default {
namespace: 'login',

state: {},

effects: {
*login({ payload }, { put, call, select }) {
const data = yield call(loginUser, payload)
if (data.success) {
yield put({ type: 'app/query' })
const { locationQuery } = yield select(_ => _.app)
const { from } = locationQuery
if (!pathMatchRegexp('/login', from)) {
if(data.group === "1") {
router.push('/user')
} else {
router.push('/dashboard')
}
} else {
router.push('/dashboard')
}
} else {
throw data
}
},
},
}
```

Файл users.js (View)

```
import React, { PureComponent } from 'react'
import PropTypes from 'prop-types'
import { router } from 'utils'
import { connect } from 'dva'
import { Row, Col, Button, Popconfirm } from 'antd'
import { withI18n } from '@lingui/react'
```

```
import { Page } from 'components'
import { stringify } from 'qs'
import List from './components/List'
import Filter from './components/Filter'
import Modal from './components/Modal'

@withI18n()
@connect(({ user, loading }) => ({ user, loading }))
class User extends PureComponent {
render() {
const { location, dispatch, user, loading, i18n } = this.props
const { query, pathname } = location
const {
list,
pagination,
currentItem,
modalVisible,
modalType,
selectedRowKeys,
} = user


return (
<Page inner>
<Filter {...filterProps} />
{selectedRowKeys.length > 0 && (
<Row style={{ marginBottom: 24, textAlign: 'right', fontSize: 13 }}>
<Col>
{`Selected ${selectedRowKeys.length} items `}
<Popconfirm
title="Are you sure delete these items?"
placement="left"
onConfirm={handleDeleteItems}
>
<Button type="primary" style={{ marginLeft: 8 }}>
Remove
</Button>
</Popconfirm>
</Col>
</Row>
)}
<List {...listProps} />
{modalVisible && <Modal {...modalProps} />}
</Page>
)
}
}

User.propTypes = {
user: PropTypes.object,
location: PropTypes.object,
dispatch: PropTypes.func,
```

```
loading: PropTypes.object,
}

export default User

import React, { PureComponent } from 'react'
import PropTypes from 'prop-types'
import { connect } from 'dva'
import { Page } from 'components'
import styles from './index.less'

@connect(({ userDetail }) => ({ userDetail }))
class UserDetail extends PureComponent {
render() {
const { userDetail } = this.props
const { data } = userDetail
const content = []
for (let key in data) {
if ({}.hasOwnProperty.call(data, key)) {
content.push(
<div key={key} className={styles.item}>
<div>{key}</div>
<div>{String(data[key])}</div>
</div>
)
}
}
return (
<Page inner>
<div className={styles.content}>{content}</div>
</Page>
)
}
}

UserDetail.propTypes = {
userDetail: PropTypes.object,
}

export default UserDetail

/* global document */
import React, { PureComponent } from 'react'
import PropTypes from 'prop-types'
import moment from 'moment'
import { FilterItem } from 'components'
import { Trans, withI18n } from '@lingui/react'
import { Form, Button, Row, Col, DatePicker, Input, Cascader } from 'antd'
import city from 'utils/city'

const { Search } = Input
const { RangePicker } = DatePicker
```

```
const ColProps = {
xs: 24,
sm: 12,
style: {
marginBottom: 16,
},
}

const TwoColProps = {
...ColProps,
xl: 96,
}

@withI18n()
@Form.create()
class Filter extends PureComponent {
handleFields = fields => {
const { createTime } = fields
if (createTime.length) {
fields.createTime = [
moment(createTime[0]).format('YYYY-MM-DD'),
moment(createTime[1]).format('YYYY-MM-DD'),
]
}
return fields
}

handleSubmit = () => {
const { onFilterChange, form } = this.props
const { getFieldsValue } = form

let fields = getFieldsValue()
fields = this.handleFields(fields)
onFilterChange(fields)
}

handleReset = () => {
const { form } = this.props
const { getFieldsValue, setFieldsValue } = form

const fields = getFieldsValue()
for (let item in fields) {
if ({}.hasOwnProperty.call(fields, item)) {
if (fields[item] instanceof Array) {
fields[item] = []
} else {
fields[item] = undefined
}
}
}
setFieldsValue(fields)
```

```
this.handleSubmit()
}
handleChange = (key, values) => {
const { form, onFilterChange } = this.props
const { getFieldsValue } = form

let fields = getFieldsValue()
fields[key] = values
fields = this.handleFields(fields)
onFilterChange(fields)
}

render() {
const { onAdd, filter, form, i18n } = this.props
const { getFieldDecorator } = form
const { name } = filter

let initialCreateTime = []
if (filter.createTime && filter.createTime[0]) {
initialCreateTime[0] = moment(filter.createTime[0])
}
if (filter.createTime && filter.createTime[1]) {
initialCreateTime[1] = moment(filter.createTime[1])
}

return (
<Row gutter={24}>
<Col {...ColProps} xl={{ span: 5 }} md={{ span: 8 }}>
{getFieldDecorator('name', { initialValue: name })(
<Search
placeholder={i18n.t`Search Name`}
onSearch={this.handleSubmit}
/>
)}
</Col>
<Col
{...ColProps}
xl={{ span: 7 }}
md={{ span: 8 }}
sm={{ span: 12 }}
id="createTimeRangePicker"
>
<FilterItem label={i18n.t`CreateTime`}>
{getFieldDecorator('createTime', {
initialValue: initialCreateTime,
})(
<RangePicker
style={{ width: '100%' }}
onChange={this.handleChange.bind(this, 'createTime')}
getCalendarContainer={() => {
return document.getElementById('createTimeRangePicker')
}}
```

```
/>
)}
</FilterItem>
</Col>
<Col
{...TwoColProps}
xl={{ span: 12 }}
md={{ span: 24 }}
sm={{ span: 24 }}
>
<Row type="flex" align="right" justify="space-between">
<div>
<Button
type="primary"
className="margin-right"
onClick={this.handleSubmit}
>
<Trans>Search</Trans>
</Button>
<Button onClick={this.handleReset}>
<Trans>Reset</Trans>
</Button>
</div>
<Button type="ghost" onClick={onAdd}>
<Trans>Create</Trans>
</Button>
</Row>
</Col>
</Row>
)
}
}

Filter.propTypes = {
onAdd: PropTypes.func,
form: PropTypes.object,
filter: PropTypes.object,
onFilterChange: PropTypes.func,
}

export default Filter


import React, { PureComponent } from 'react'
import PropTypes from 'prop-types'
import { Table, Modal } from 'antd'
import { DropOption } from 'components'
import { Trans, withI18n } from '@lingui/react'
import Link from 'umi/link'
import styles from './List.less'

const { confirm } = Modal
```

```
@withI18n()
class List extends PureComponent {
handleMenuClick = (record, e) => {
const { onDeleteItem, onEditItem, i18n } = this.props

if (e.key === '1') {
onEditItem(record)
} else if (e.key === '2') {
confirm({
title: i18n.t`Are you sure delete this record?`,
onOk() {
onDeleteItem(record.id)
},
})
}
}

render() {
const { onDeleteItem, onEditItem, i18n, ...tableProps } = this.props

const columns = [
{
title: <Trans>NickName</Trans>,
dataIndex: 'nickName',
key: 'nickName',
width: 72,
fixed: 'left',
render: (text, record) => <Link to={`user/${record.id}`}>{text}</Link>,
},
{
title: <Trans>Name</Trans>,
dataIndex: 'f_name',
key: 'f_name',
render: (text, record) => <Link to={`user/${record.id}`}>{text}</Link>,
},
{
title: <Trans>F_name</Trans>,
dataIndex: 's_name',
key: 's_name',
render: (text, record) => <Link to={`user/${record.id}`}>{text}</Link>,
},
{
title: <Trans>L_name</Trans>,
dataIndex: 'l_name',
key: 'l_name',
render: (text, record) => <Link to={`user/${record.id}`}>{text}</Link>,
},
{
title: <Trans>Group</Trans>,
dataIndex: 'group',
key: 'group',
```

```
render: (text, record) => <Link to={`user/${record.id}`}>{text}</Link>,
},
{
title: <Trans>CreateTime</Trans>,
dataIndex: 'createTime',
key: 'createTime',
},
{
title: <Trans>Operation</Trans>,
key: 'operation',
fixed: 'right',
render: (text, record) => {
return (
<DropOption
onMenuClick={e => this.handleMenuClick(record, e)}
menuOptions={[
{ key: '1', name: i18n.t`Update` },
{ key: '2', name: i18n.t`Delete` },
]}
/>
)
},
},
]

return (
<Table
{...tableProps}
pagination={{
...tableProps.pagination,
showTotal: total => i18n.t`Total ${total} Items`,
}}
className={styles.table}
bordered
scroll={{ x: 1200 }}
columns={columns}
simple
rowKey={record => record.id}
/>
)
}
}

List.propTypes = {
onDeleteItem: PropTypes.func,
onEditItem: PropTypes.func,
location: PropTypes.object,
}

export default List
```

```
import React, { PureComponent } from 'react'
import PropTypes from 'prop-types'
import { Form, Input, Checkbox , Modal, Select, Layout } from 'antd'
import { withI18n } from '@lingui/react'

const FormItem = Form.Item

const {
Header, Content,
} = Layout

const CheckboxGroup = Checkbox.Group;

const plainOptions = ['Apple', 'Pear', 'Orange'];

const formItemLayout = {
labelCol: {
span: 6,
},
wrapperCol: {
span: 14,
}
}
@withI18n()
@Form.create()
class UserModal extends PureComponent {
state = {
group: null,
checkedList: null,
indeterminate: true,
checkAll: false,
}

handleOk = () => {
const { item = {}, onOk, form } = this.props
const { validateFields, getFieldsValue } = form

validateFields(errors => {
if (errors) {
return
}
const data = {
...getFieldsValue(),
key: item.key,
}

onOk(data)
})
}

handleSelect = (val) => {
this.setState({
```

```
group: val
})
}

onChange = (checkedList) => {
this.setState({
checkedList,
indeterminate: !!checkedList.length && (checkedList.length < plainOptions.length),
checkAll: checkedList.length === plainOptions.length,
});
}

onCheckAllChange = (e) => {
this.setState({
checkedList: e.target.checked ? plainOptions : [],
indeterminate: false,
checkAll: e.target.checked,
});
}

render() {
const { item = {}, onOk, form, i18n, ...modalProps } = this.props
const { getFieldDecorator } = form

return (
<Modal {...modalProps} onOk={this.handleOk}>
<Form layout="horizontal">
<FormItem label={i18n.t`Name`} hasFeedback {...formItemLayout}>
{getFieldDecorator('name', {
initialValue: item.f_name,
rules: [
{
required: true,
},
],
})(<Input />)}
</FormItem>
<FormItem label={i18n.t`F_name`} hasFeedback {...formItemLayout}>
{getFieldDecorator('s_name', {
initialValue: item.s_name,
rules: [
{
required: true,
},
],
})(<Input />)}
</FormItem>
<FormItem label={i18n.t`L_name`} hasFeedback {...formItemLayout}>
{getFieldDecorator('f_name', {
initialValue: item.f_name,
rules: [
{
```

```
required: true,
},
],
})(<Input />)}
</FormItem>

<FormItem label={i18n.t`NickName`} hasFeedback {...formItemLayout}>
{getFieldDecorator('nickName', {
initialValue: item.nickName,
rules: [
{
required: true,
},
],
})(<Input />)}
</FormItem>
<FormItem label={i18n.t`Password`} hasFeedback {...formItemLayout}>
{getFieldDecorator('password', {
initialValue: item.password,
rules: [
{
required: true,
},
],
})(<Input />)}
</FormItem>
<FormItem label={i18n.t`Group`} hasFeedback {...formItemLayout}>
{getFieldDecorator('group', {
rules: [
{
required: true,
},
],
initialValue: item.group,
})(
<Select
placeholder="Выбрать группу пользователя"
onChange={this.handleSelect}
>
<Select.Option value="2">Менеджер</Select.Option>
<Select.Option value="1">Администратор</Select.Option>
</Select>
)}
</FormItem>
{
this.state.group && this.state.group === "manager" ? (
<FormItem >
<Layout style={{
backgroundColor: "transparent"
}}>
<Header style={{
backgroundColor: "transparent",
```

```
textAlign: 'center',
fontSize: 20
}}>
{i18n.t`Select category`}
</Header>
<Content>
<Checkbox
indeterminate={this.state.indeterminate}
onChange={this.onCheckAllChange}
checked={this.state.checkAll}
>
Выбрать всё
</Checkbox>
<br />
{getFieldDecorator('category', {
initialValue: item.category,
rules: [
{
required: true,
},
],
})(
<CheckboxGroup options={plainOptions} onChange={this.onChange} />
)}
</Content>
</Layout>
</FormItem>
) : null

}
</Form>
</Modal>
)
}
}

UserModal.propTypes = {
type: PropTypes.string,
item: PropTypes.object,
onOk: PropTypes.func,
}

export default UserModal
```

Файл cusers.js (Controllers)

```
const handleRefresh = newQuery => {
router.push({
```

```
pathname,
search: stringify(
{
...query,
...newQuery,
},
{ arrayFormat: 'repeat' }
),
})
}

const modalProps = {
item: modalType === 'create' ? {} : currentItem,
visible: modalVisible,
maskClosable: false,
confirmLoading: loading.effects[`user/${modalType}`],
title: `${
modalType === 'create' ? i18n.t`Create User` : i18n.t`Update User`
}`,
centered: true,
onOk(data) {
dispatch({
type: `user/${modalType}`,
payload: data,
}).then(() => {
handleRefresh()
})
},
onCancel() {
dispatch({
type: 'user/hideModal',
})
},
}

const listProps = {
dataSource: list,
loading: loading.effects['user/query'],
pagination,
onChange(page) {
handleRefresh({
page: page.current,
pageSize: page.pageSize,
})
},
onDeleteItem(id) {
dispatch({
type: 'user/delete',
payload: id,
}).then(() => {
handleRefresh({
page:
```

```
list.length === 1 && pagination.current > 1
? pagination.current - 1
: pagination.current,
})
})
},
onEditItem(item) {
dispatch({
type: 'user/showModal',
payload: {
modalType: 'update',
currentItem: item,
},
})
},
rowSelection: {
selectedRowKeys,
onChange: keys => {
dispatch({
type: 'user/updateState',
payload: {
selectedRowKeys: keys,
},
})
},
},
}

const filterProps = {
filter: {
...query,
},
onFilterChange(value) {
handleRefresh({
...value,
page: 1,
})
},
onAdd() {
dispatch({
type: 'user/showModal',
payload: {
modalType: 'create',
},
})
},
}

const handleDeleteItems = () => {
dispatch({
type: 'user/multiDelete',
payload: {
```

```
ids: selectedRowKeys,
},
}).then(() => {
handleRefresh({
page:
list.length === selectedRowKeys.length && pagination.current > 1
? pagination.current - 1
: pagination.current,
})
})
}
```

Файл cusers.js (Models)

```
/* global window */
import modelExtend from 'dva-model-extend'
import { pathMatchRegexp } from 'utils'
import api from 'api'
import { pageModel } from 'utils/model'

const {
queryUserList,
createUser,
removeUser,
updateUser,
removeUserList,
} = api

export default modelExtend(pageModel, {
namespace: 'user',

state: {
currentItem: {},
modalVisible: false,
modalType: 'create',
selectedRowKeys: [],
},

subscriptions: {
setup({ dispatch, history }) {
history.listen(location => {
if (pathMatchRegexp('/user', location.pathname)) {
const payload = location.query || { page: 1, pageSize: 10 }
dispatch({
type: 'query',
payload,
})
}
})
},
```

```
},

effects: {
*query({ payload = {} }, { call, put }) {
const data = yield call(queryUserList, payload)
if (data) {
yield put({
type: 'querySuccess',
payload: {
list: data.data,
pagination: {
current: Number(payload.page) || 1,
pageSize: Number(payload.pageSize) || 10,
total: data.total,
},
},
})
}
},

*delete({ payload }, { call, put, select }) {
const data = yield call(removeUser, { id: payload })
const { selectedRowKeys } = yield select(_ => _.user)
if (data.success) {
yield put({
type: 'updateState',
payload: {
selectedRowKeys: selectedRowKeys.filter(_ => _ !== payload),
},
})
} else {
throw data
}
},

*multiDelete({ payload }, { call, put }) {
const data = yield call(removeUserList, payload)
if (data.success) {
yield put({ type: 'updateState', payload: { selectedRowKeys: [] } })
} else {
throw data
}
},

*create({ payload }, { call, put }) {
const data = yield call(createUser, payload)
if (data.success) {
yield put({ type: 'hideModal' })
} else {
throw data
}
},
```

```
*update({ payload }, { select, call, put }) {
const id = yield select(({ user }) => user.currentItem.id)
const newUser = { ...payload, id }
const data = yield call(updateUser, newUser)
if (data.success) {
yield put({ type: 'hideModal' })
} else {
throw data
}
},
},

reducers: {
showModal(state, { payload }) {
return { ...state, ...payload, modalVisible: true }
},

hideModal(state) {
return { ...state, modalVisible: false }
},
},
})


import { pathMatchRegexp } from 'utils'
import api from 'api'

const { queryUser } = api

export default {
namespace: 'userDetail',

state: {
data: {},
},

subscriptions: {
setup({ dispatch, history }) {
history.listen(({ pathname }) => {
const match = pathMatchRegexp('/user/:id', pathname)
if (match) {
dispatch({ type: 'query', payload: { id: match[1] } })
}
})
},
},

effects: {
*query({ payload }, { call, put }) {
const data = yield call(queryUser, payload)
const { success, message, status, ...other } = data
```

```
if (success) {
yield put({
type: 'querySuccess',
payload: {
data: other,
},
})
} else {
throw data
}
},
},

reducers: {
querySuccess(state, { payload }) {
const { data } = payload
return {
...state,
data,
}
},
},
}
```

Файл Dashboard.js (View)

```
import React, { PureComponent } from 'react'
import PropTypes from 'prop-types'
import { connect } from 'dva'
import { Row, Col, Card, Icon } from 'antd'

import { Page } from 'components'

import styles from './index.less'

const Color = {
green: '#64ea91',
blue: '#8fc9fb',
purple: '#d897eb',
red: '#f69899',
yellow: '#f8c82e',
peach: '#f797d6',
borderBase: '#e5e5e5',
borderSplit: '#f4f4f4',
grass: '#d6fbb5',
sky: '#c1e0fc',
}
```

```
const statuses = [
{
title: "Расписание успешно добавлено",
color: Color.green
},
{
title: "Расписание успешно изменено",
color: Color.yellow
},
{
title: "Расписание успешно удалено",
color: Color.red
}
]

@connect(({ app, dashboard, loading }) => ({
username: app.user.username,
dashboard,
loading,
}))
class Dashboard extends PureComponent {
render() {
const { dashboard } = this.props
const {
data,
} = dashboard

const numberCards = () => data.map((item, key) => (
<Col key={key} lg={6} md={12}>
<Card
title={`Заказ #${item.id}`}
style={{minHeight: 180, backgroundColor: statuses[item.status -1].color}}
actions={[<Icon type="eye" />, <Icon type="edit" />]}
>
<p style={{ textAlign: 'center' }}>{statuses[item.status -1].title}</p>
<p style={{ paddingTop: 15 }}>{item.date}</p>

</Card>
</Col>
))

return (
<Page
className={styles.dashboard}
>
<Row gutter={24}>
{data && numberCards()}
</Row>
</Page>
)
}
}
```

```
Dashboard.propTypes = {
username: PropTypes.string,
dashboard: PropTypes.object,
loading: PropTypes.object,
}

export default Dashboard

import React from 'react'
import PropTypes from 'prop-types'
import { Icon, Card } from 'antd'
import styles from './numberCard.less'

function NumberCard({ icon, color, title, status, date, countUp }) {
return (
<Card
className={styles.numberCard}
bordered={false}
bodyStyle={{ padding: 10 }}
>
<Icon className={styles.iconWarp} style={{ color }} type={icon} />
<div className={styles.content}>
<p className={styles.title}>{title || 'No Title'}</p>
<p className={styles.title}>
{status}
</p>
<p className={styles.title}>
{date}
</p>
</div>
</Card>
)
}

NumberCard.propTypes = {
icon: PropTypes.string,
color: PropTypes.string,
title: PropTypes.string,
status: PropTypes.string,
date: PropTypes.string,
countUp: PropTypes.object,
}

export default NumberCard
```

Файл Dashboard.js (Controllers)

```
import { request, config } from 'utils'

const { api } = config
const { dashboard } = api

export function query(params) {
return request({
url: dashboard,
withCredentials: true,
method: 'get',
data: params,
})
}
```

Файл Dashboard.js (Model)

```
import { parse } from 'qs'
import modelExtend from 'dva-model-extend'
import api from 'api'
import { pathMatchRegexp } from 'utils'
import { model } from 'utils/model'

const { queryDashboard } = api

export default modelExtend(model, {
namespace: 'dashboard',
state: {
sales: [],
numbers: [],
recentSales: [],
user: {},
subscriptions: {
setup({ dispatch, history }) {
history.listen(({ pathname }) => {
if (
pathMatchRegexp('/dashboard', pathname) ||
pathMatchRegexp('/', pathname)
) {
dispatch({ type: 'query' })
}
})
},
},
effects: {
*query({ payload }, { call, put }) {
const data = yield call(queryDashboard, parse(payload))
yield put({
type: 'updateState',
payload: data,
```

```
})
},
},
})
```

Файл Shedule.js (Model)

```
import React, { PureComponent } from 'react'
import PropTypes from 'prop-types'
import { connect } from 'dva'
import { Row, Col, Card, Icon, Pagination } from 'antd'

import { Page } from 'components'
import { router } from 'utils'
import { stringify } from 'qs'
import { withI18n } from '@lingui/react'

@withI18n()
@connect(({ events, loading }) => ({ events, loading }))
class Events extends PureComponent {

render() {
const { events, loading, location, dispatch, i18n } = this.props
const { list } = events

const eventCards = list.map((item, key) => (
<Col key={key} lg={6} md={12}>
<Card
title={item.Name}
style={{minHeight: 180, marginBottom: 30 }}
actions={[<Icon type="plus-circle" onClick={ () => handleClick({event_id: item.id_event,
count: 1}) }/>]}
>
<p style={{ }}>{`Стоимость: ${item.price} руб. за 1 шт.`}</p>
<p style={{ paddingTop: 15 }}>{`На складе осталось: ${item.count} шт.`}</p>

</Card>
</Col>
))

return (
<Page
>
<Row gutter={24}>
{eventCards}
</Row>
</Page>
)
}
```

```
}

Events.propTypes = {
events: PropTypes.object,
loading: PropTypes.object,
location: PropTypes.object,
dispatch: PropTypes.func,
}

export default Events

import React, { PureComponent } from 'react'
import { Table, Avatar } from 'antd'
import { withI18n } from '@lingui/react'
import { Ellipsis } from 'ant-design-pro'
import styles from './List.less'

@withI18n()
class List extends PureComponent {
render() {
const { i18n, ...tableProps } = this.props
const columns = [
{
title: i18n.t`Image`,
dataIndex: 'image',
render: text => <Avatar shape="square" src={text} />,
},
{
title: i18n.t`Title`,
dataIndex: 'title',
render: text => (
<Ellipsis tooltip length={30}>
{text}
</Ellipsis>
),
},
{
title: i18n.t`Author`,
dataIndex: 'author',
},
{
title: i18n.t`Categories`,
dataIndex: 'categories',
},
{
title: i18n.t`Tags`,
dataIndex: 'tags',
},
{
title: i18n.t`Visibility`,
dataIndex: 'visibility',
},
```

```
{
title: i18n.t`Comments`,
dataIndex: 'comments',
},
{
title: i18n.t`Views`,
dataIndex: 'views',
},
{
title: i18n.t`Publish Date`,
dataIndex: 'date',
},
]

return (
<Table
{...tableProps}
pagination={{
...tableProps.pagination,
showTotal: total => i18n.t`Total ${total} Items`,
}}
bordered
scroll={{ x: 1200 }}
className={styles.table}
columns={columns}
simple
rowKey={record => record.id}
/>
)
}
}

export default List
```

Файл cShedule.js (Controllers)

```
const handleClick = data => {
dispatch({
type: `events/create`,
payload: data,
}).then(() => {

})
}

handleChange = (current,pageSize) => {
const {
location
} = this.props
```

```
const {
query,
pathname
} = location

router.push({
pathname,
search: stringify({
...query,
page: current,
pageSize: pageSize,
}),
})
}
```

Файл mShedule.js (Model)

```
import modelExtend from 'dva-model-extend'
import api from 'api'
import { pathMatchRegexp } from 'utils'
import { pageModel } from 'utils/model'

const {
queryEventList,
createEvent,
} = api

export default modelExtend(pageModel, {
namespace: 'events',

subscriptions: {
setup({ dispatch, history }) {
history.listen(location => {
if (pathMatchRegexp('/events', location.pathname)) {
dispatch({
type: 'query',
payload: {
...location.query,
},
})
}
})
},
},

effects: {
*query({ payload }, { call, put }) {
const data = yield call(queryEventList, payload)
if (data.success) {
yield put({
```

```
type: 'querySuccess',
payload: {
list: data.data,
pagination: {
current: Number(payload.page) || 1,
pageSize: Number(payload.pageSize) || 10,
total: data.total,
},
},
})
} else {
throw data
}
},

*create({ payload }, { call, put }) {
const data = yield call(createEvent, payload)
if (data.success) {

} else {
throw data
}
},
},
})
```

Файл QEvent.js (View)

```
import React, { PureComponent } from 'react'
import PropTypes from 'prop-types'
import { router } from 'utils'
import { connect } from 'dva'
import { Row, Col, Button, Popconfirm } from 'antd'
import { withI18n } from '@lingui/react'
import { Page } from 'components'
import { stringify } from 'qs'
import List from './components/List'
import Filter from './components/Filter'

@withI18n()
@connect(({ shedules, loading }) => ({ shedules, loading }))
class Shedules extends PureComponent {
render() {
const { location, dispatch, shedules, loading, i18n } = this.props
const { query, pathname } = location
const {
list,
pagination,
selectedRowKeys,
} = shedules
```

```
return (
<Page inner>
<Filter {...filterProps} />
{selectedRowKeys.length > 0 && (
<Row style={{ marginBottom: 24, textAlign: 'right', fontSize: 13 }}>
<Col>
{`Выбрано ${selectedRowKeys.length} записей `}
<Popconfirm
title="Вы уверены, что хотите удалить?"
placement="left"
onConfirm={handleDeleteItems}
>
<Button type="primary" style={{ marginLeft: 8 }}>
Удалить
</Button>
</Popconfirm>
</Col>
</Row>
)}
<List {...listProps} />
</Page>
)
}
}

Shedules.propTypes = {
shedules: PropTypes.object,
location: PropTypes.object,
dispatch: PropTypes.func,
loading: PropTypes.object,
}

export default Shedules

import React, { PureComponent } from 'react'
import PropTypes from 'prop-types'
import moment from 'moment'
import { FilterItem } from 'components'
import { Trans, withI18n } from '@lingui/react'
import { Form, Button, Row, Col, DatePicker, Input, Cascader } from 'antd'
import city from 'utils/city'

const { Search } = Input
const { RangePicker } = DatePicker

const ColProps = {
xs: 24,
sm: 12,
style: {
marginBottom: 16,
```

```
},
}

const TwoColProps = {
...ColProps,
xl: 96,
}

@withI18n()
@Form.create()
class Filter extends PureComponent {
handleFields = fields => {
const { createTime } = fields
if (createTime.length) {
fields.createTime = [
moment(createTime[0]).format('YYYY-MM-DD'),
moment(createTime[1]).format('YYYY-MM-DD'),
]
}
return fields
}

handleSubmit = () => {
const { onFilterChange, form } = this.props
const { getFieldsValue } = form

let fields = getFieldsValue()
fields = this.handleFields(fields)
onFilterChange(fields)
}

handleReset = () => {
const { form } = this.props
const { getFieldsValue, setFieldsValue } = form

const fields = getFieldsValue()
for (let item in fields) {
if ({}.hasOwnProperty.call(fields, item)) {
if (fields[item] instanceof Array) {
fields[item] = []
} else {
fields[item] = undefined
}
}
}
setFieldsValue(fields)
this.handleSubmit()
}
handleChange = (key, values) => {
const { form, onFilterChange } = this.props
const { getFieldsValue } = form
```

```
let fields = getFieldsValue()
fields[key] = values
fields = this.handleFields(fields)
onFilterChange(fields)
}

render() {
const { onAdd, filter, form, i18n } = this.props
const { getFieldDecorator } = form
const { name } = filter

let initialCreateTime = []
if (filter.createTime && filter.createTime[0]) {
initialCreateTime[0] = moment(filter.createTime[0])
}
if (filter.createTime && filter.createTime[1]) {
initialCreateTime[1] = moment(filter.createTime[1])
}

return (
<Row gutter={24}>
<Col {...ColProps} xl={{ span: 5 }} md={{ span: 8 }}>
{getFieldDecorator('name', { initialValue: name })(
<Search
placeholder={i18n.t`Search Name`}
onSearch={this.handleSubmit}
/>
)}
</Col>
<Col
{...ColProps}
xl={{ span: 7 }}
md={{ span: 8 }}
sm={{ span: 12 }}
id="createTimeRangePicker"
>
<FilterItem label={i18n.t`CreateTime`}>
{getFieldDecorator('createTime', {
initialValue: initialCreateTime,
})(
<RangePicker
style={{ width: '100%' }}
onChange={this.handleChange.bind(this, 'createTime')}
getCalendarContainer={() => {
return document.getElementById('createTimeRangePicker')
}}
/>
)}
</FilterItem>
</Col>
<Col
{...TwoColProps}
```

```
xl={{ span: 12 }}
md={{ span: 24 }}
sm={{ span: 24 }}
>
<Row type="flex" align="right" justify="space-between">
<div>
<Button
type="primary"
className="margin-right"
onClick={this.handleSubmit}
>
<Trans>Search</Trans>
</Button>
<Button onClick={this.handleReset}>
<Trans>Reset</Trans>
</Button>
</div>
<Button type="ghost" onClick={onAdd}>
<Trans>Create</Trans>
</Button>
</Row>
</Col>
</Row>
)
}
}

Filter.propTypes = {
onAdd: PropTypes.func,
form: PropTypes.object,
filter: PropTypes.object,
onFilterChange: PropTypes.func,
}

export default Filter

import React, { PureComponent } from 'react'
import PropTypes from 'prop-types'
import { Table, Modal } from 'antd'
import { DropOption } from 'components'
import { Trans, withI18n } from '@lingui/react'
import Link from 'umi/link'
import styles from './List.less'

const { confirm } = Modal

@withI18n()
class List extends PureComponent {
handleMenuClick = (record, e) => {
const { onDeleteItem, onDownloadItem, i18n } = this.props

if (e.key === '1') {
```

```
onDownloadItem(record)
} else if (e.key === '2') {
confirm({
title: i18n.t`Are you sure delete this record?`,
onOk() {
onDeleteItem(record.id)
},
})
}
}

render() {
const { onDeleteItem, onDownloadItem, i18n, ...tableProps } = this.props

const columns = [
{
title: <Trans>Title</Trans>,
dataIndex: 'name',
key: 'name',
},
{
title: <Trans>CreateTime</Trans>,
dataIndex: 'createDate',
key: 'createDate',
},
{
title: <Trans>Operation</Trans>,
key: 'operation',
fixed: 'right',
render: (text, record) => {
return (
<DropOption
onMenuClick={e => this.handleMenuClick(record, e)}
menuOptions={[
{ key: '1', name: i18n.t`Download` },
{ key: '2', name: i18n.t`Delete` },
]}
/>
)
},
},
]

return (
<Table
{...tableProps}

className={styles.table}
bordered
columns={columns}
simple
rowKey={record => record.id}
```

```
/>
)
}
}

List.propTypes = {
onDeleteItem: PropTypes.func,
onEditItem: PropTypes.func,
location: PropTypes.object,
}

export default List

import React, { PureComponent } from 'react'
import PropTypes from 'prop-types'
import { connect } from 'dva'
import { Page } from 'components'
import styles from './index.less'

@connect(({ sheduleDetail }) => ({ sheduleDetail }))
class SheduleDetail extends PureComponent {
render() {
const { sheduleDetail } = this.props
const { data } = sheduleDetail
const content = []
for (let key in data) {
if ({}.hasOwnProperty.call(data, key)) {
content.push(
<div key={key} className={styles.item}>
<div>{key}</div>
<div>{String(data[key])}</div>
</div>
)
}
}
return (
<Page inner>
<div className={styles.content}>{content}</div>
</Page>
)
}
}

SheduleDetail.propTypes = {
sheduleDetail: PropTypes.object,
}

export default SheduleDetail
```

Файл cEvent.js (Controllers)

```
const handleRefresh = newQuery => {
router.push({
pathname,
search: stringify(
{
...query,
...newQuery,
},
{ arrayFormat: 'repeat' }
),
})
}

const listProps = {
dataSource: list.data,
loading: loading.effects['shedules/query'],
onDownloadItem(item) {
window.open(`${document.location.origin}/${item.path}/${item.name}`, '_blank')
},
onDeleteItem(id) {
dispatch({
type: 'shedules/delete',
payload: id,
}).then(() => {
handleRefresh()
})
},
rowSelection: {
selectedRowKeys,
onChange: keys => {
dispatch({
type: 'shedules/updateState',
payload: {
selectedRowKeys: keys,
},
})
},
},
}

const filterProps = {
filter: {
...query,
},
onFilterChange(value) {
handleRefresh({
...value,
page: 1,
})
```

```
},
onAdd(data) {
dispatch({
type: `shedules/create`,
payload: data,
}).then(() => {
handleRefresh()
})
},
}

const handleDeleteItems = () => {
dispatch({
type: 'shedules/multiDelete',
payload: {
ids: selectedRowKeys,
},
}).then(() => {
handleRefresh({
page:
list.length === selectedRowKeys.length && pagination.current > 1
? pagination.current - 1
: pagination.current,
})
})
}
```

Файл cEvent.js (Models)

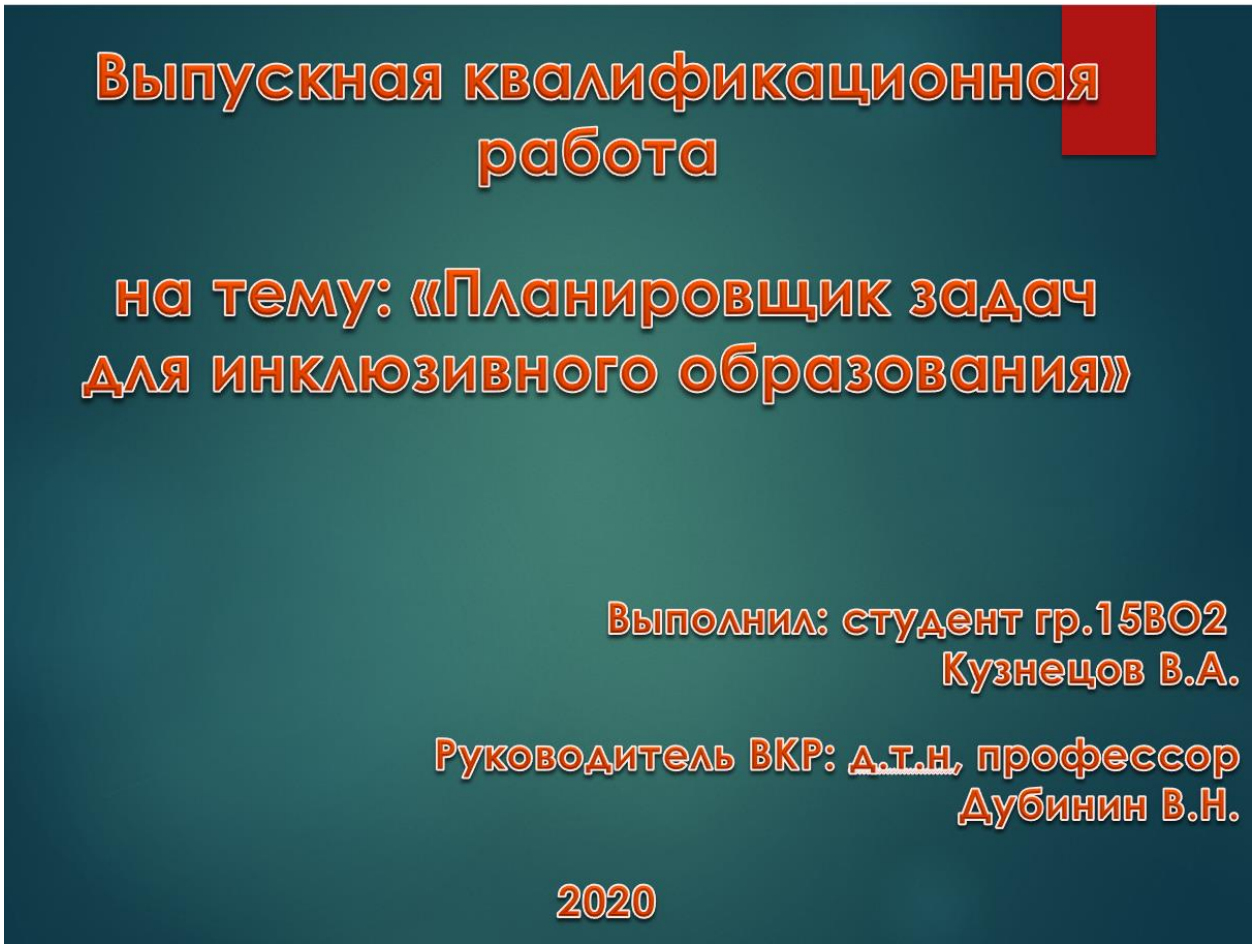
```
/* global window */
import modelExtend from 'dva-model-extend'
import { pathMatchRegexp } from 'utils'
import api from 'api'
import { pageModel } from 'utils/model'

const {
queryShedules,
createShedule,
removeShedule,
removeShedulesList,
} = api

export default modelExtend(pageModel, {
namespace: 'shedules',

state: {
currentItem: {},
modalVisible: false,
modalType: 'create',
selectedRowKeys: [],
},
```

```
subscriptions: {
setup({ dispatch, history }) {
history.listen(location => {
if (pathMatchRegexp('/shedules', location.pathname)) {
const payload = location.query || { page: 1, pageSize: 10 }
dispatch({
type: 'query',
payload,
})
}
})
},
},

effects: {
*query({ payload = {} }, { call, put }) {
const data = yield call(queryShedules, payload)
if (data) {
yield put({
type: 'querySuccess',
payload: {
list: data,
pagination: {
current: Number(payload.page) || 1,
pageSize: Number(payload.pageSize) || 10,
total: data.total,
},
},
})
}
},

*delete({ payload }, { call, put, select }) {
const data = yield call(removeShedule, { id: payload })
const { selectedRowKeys } = yield select(_ => _.user)
if (data.success) {
yield put({
type: 'updateState',
payload: {
selectedRowKeys: selectedRowKeys.filter(_ => _ !== payload),
},
})
} else {
throw data
}
},

*multiDelete({ payload }, { call, put }) {
const data = yield call(removeShedulesList, payload)
if (data.success) {
yield put({ type: 'updateState', payload: { selectedRowKeys: [] } })
```

```
} else {
throw data
}
},

*create({ payload }, { call, put }) {
const data = yield call(createShedule, payload)
if (data.success) {
yield put({ type: 'hideModal' })
} else {
throw data
}
},
},

reducers: {
showModal(state, { payload }) {
return { ...state, ...payload, modalVisible: true }
},

hideModal(state) {
return { ...state, modalVisible: false }
},
},
})

import { pathMatchRegexp } from 'utils'
import api from 'api'

const { queryShedules } = api

export default {
namespace: 'sheduleDetail',

state: {
data: {},
},

subscriptions: {
setup({ dispatch, history }) {
history.listen(({ pathname }) => {
const match = pathMatchRegexp('/shedule/:id', pathname)
if (match) {
dispatch({ type: 'query', payload: { id: match[1] } })
}
})
},
},

effects: {
*query({ payload }, { call, put }) {
const data = yield call(queryShedules, payload)
```

```
const { success, message, status, ...other } = data
if (success) {
yield put({
type: 'querySuccess',
payload: {
data: other,
},
})
} else {
throw data
}
},
},

reducers: {
querySuccess(state, { payload }) {
const { data } = payload
return {
...state,
data,
}
},
},
}
```

Презентация
Приложение В
(обязательное)



Целью работы является разработка планировщика задач, отвечающей требованиям, необходимых для выполнения составления планов мероприятий и учебного процесса для классов и учителей, а также позволяющей хранить большой объем информации и получать доступ к ним по средствам сети интернет.

Объектом исследования является планировщик задач.

Система работает следующим образом: у каждого учителя и ученика есть свой логин и пароль, который регистрирует им администратор. Учителя получают доступ к данным в web-приложении и могут их просматривать и редактировать. Ученики также получают доступ и могут только их просматривать. С помощью календаря можно выбрать нужный день. Ученики получают оповещения о начале занятия или мероприятия.

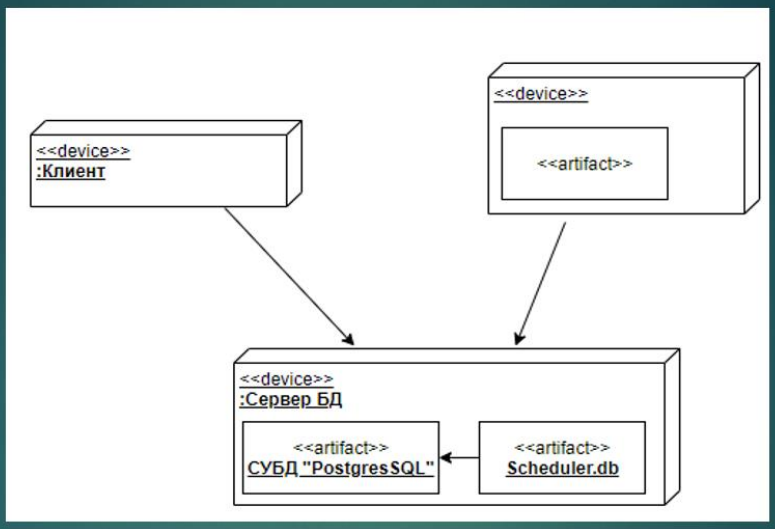


Диаграмма пакетов

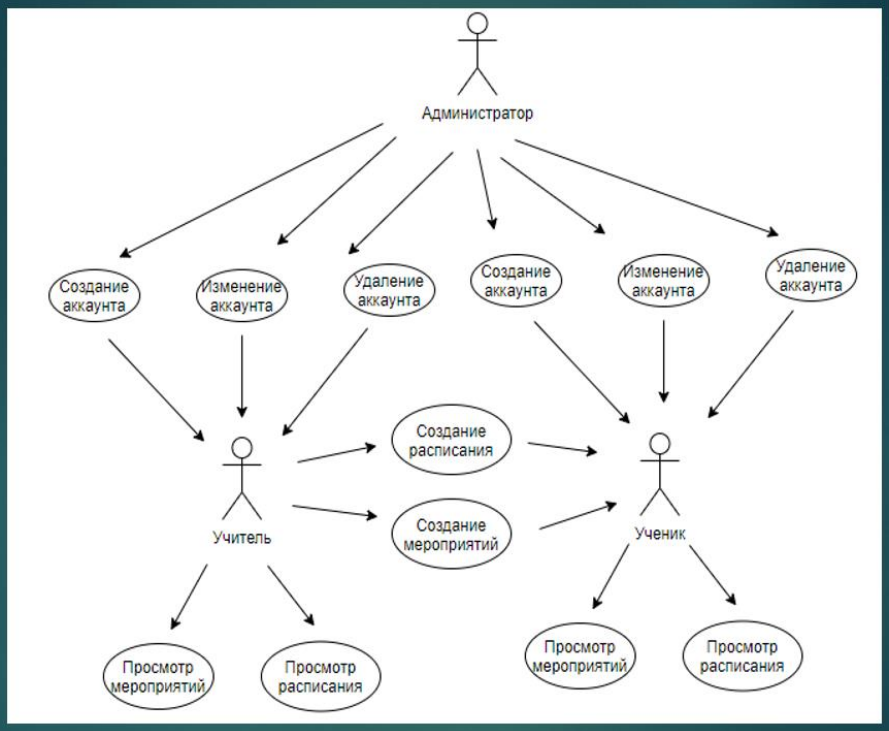


Диаграмма вариантов использования

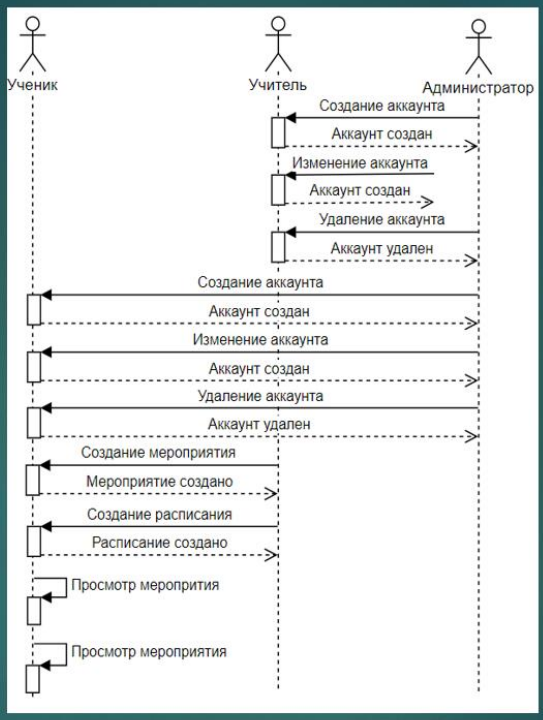
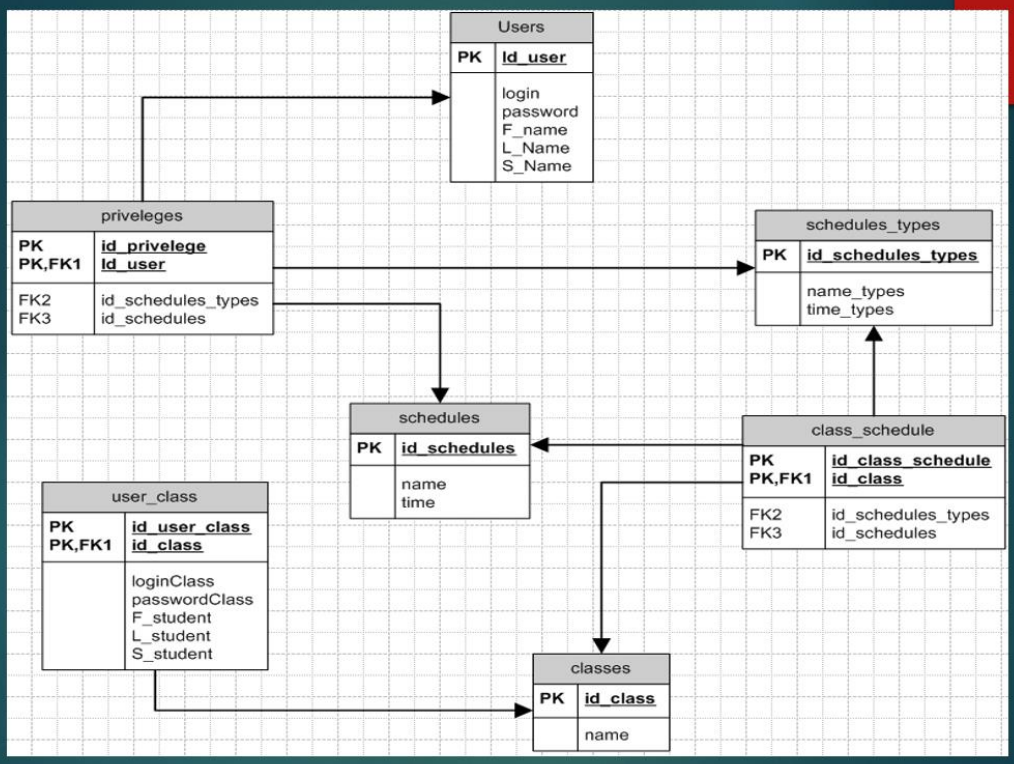
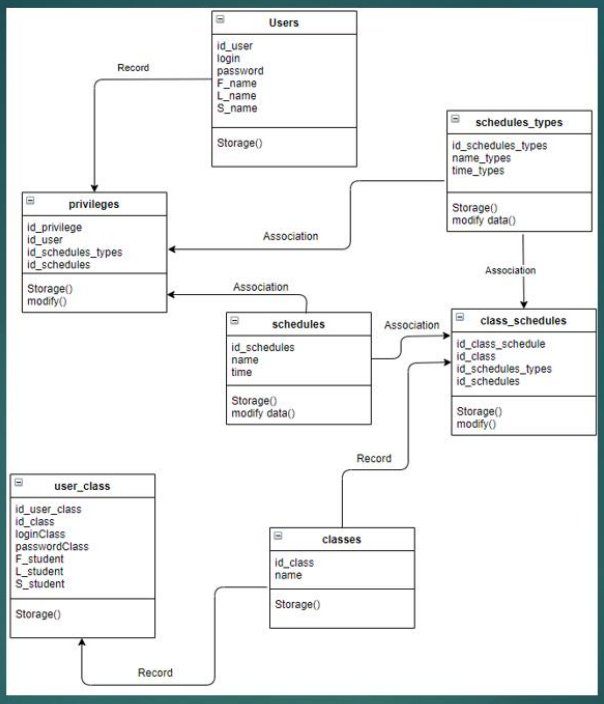


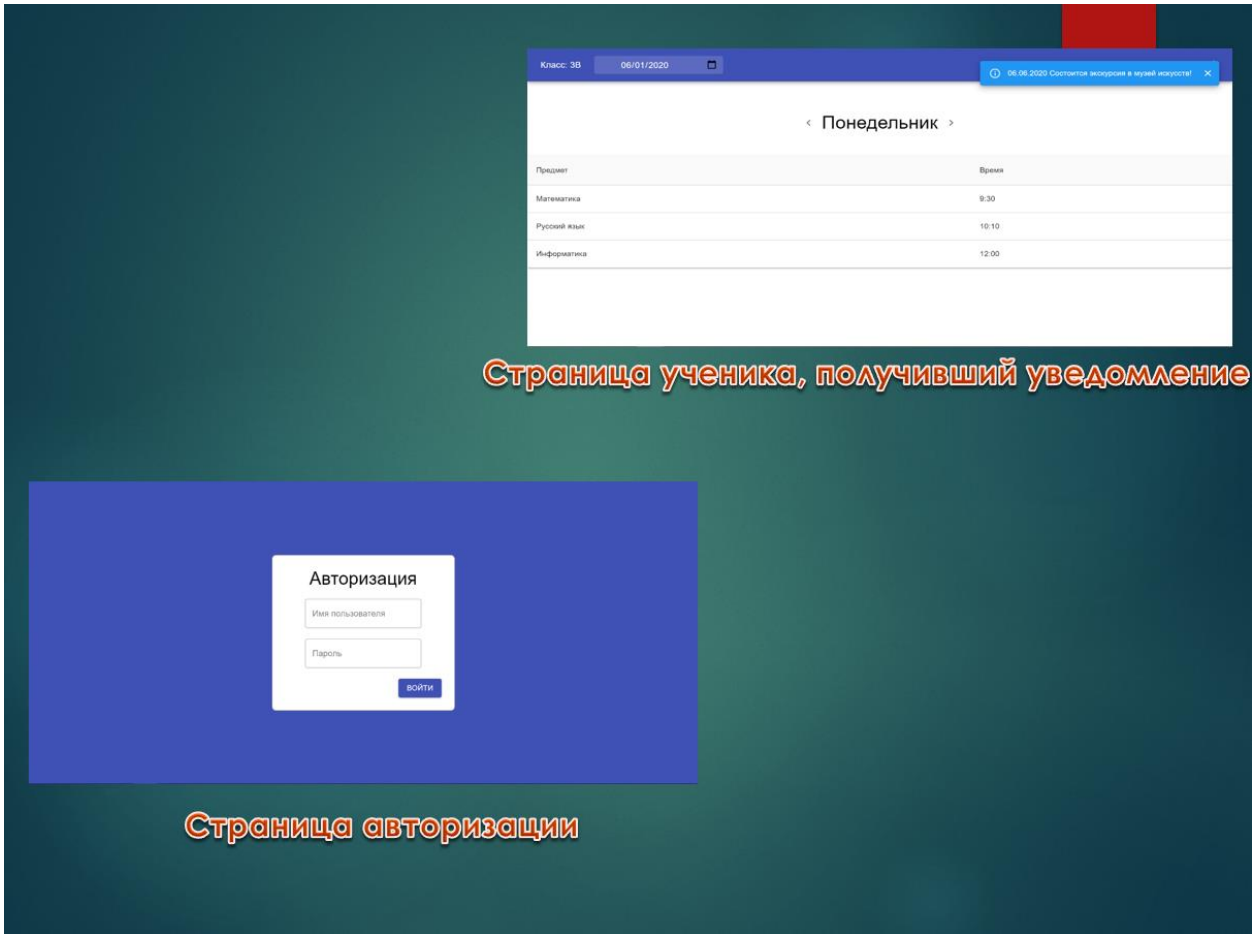
Диаграмма последовательностей

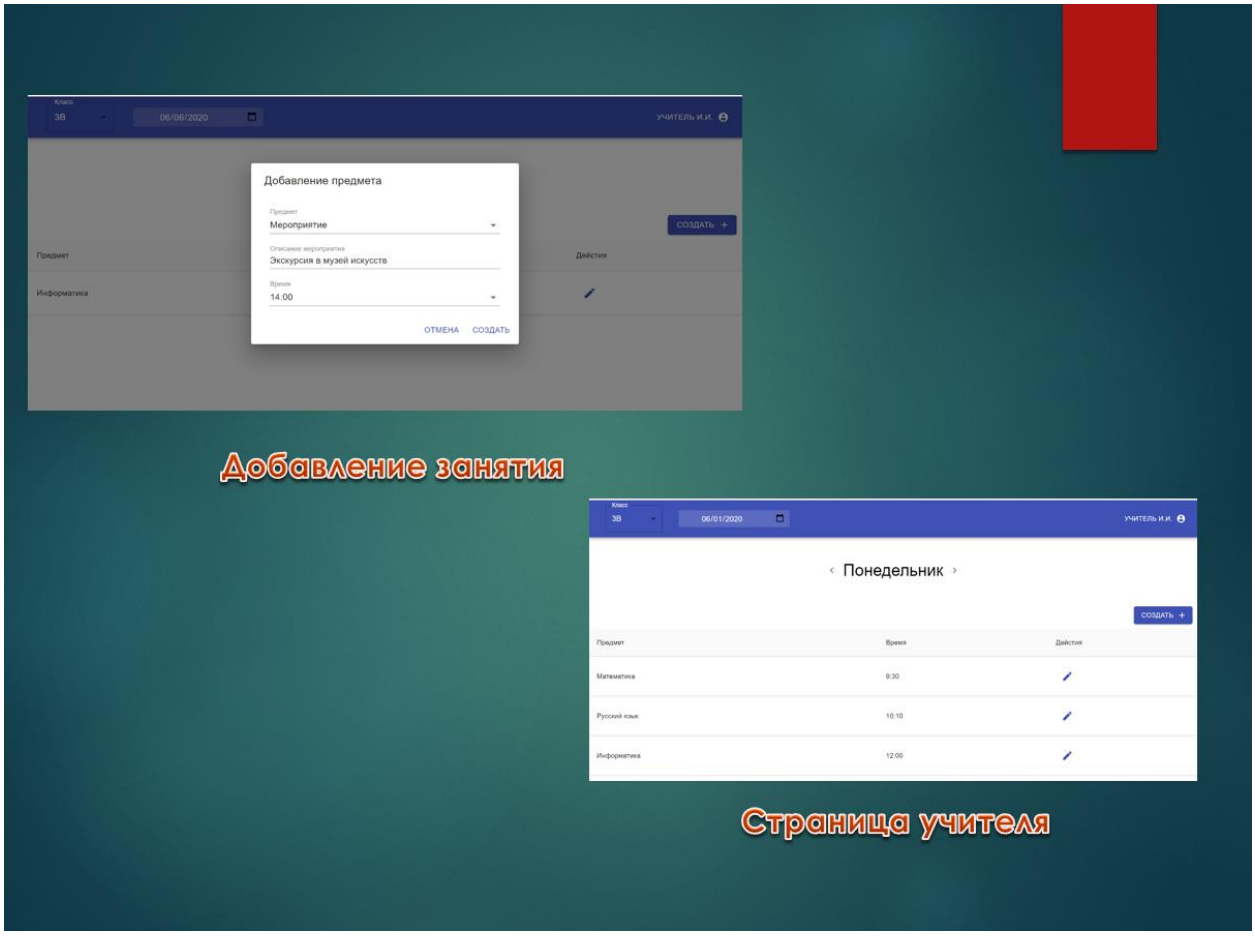


База данных



Добавление занятия


Класс: 3В
06/01/2020
06.06.2020 Состоится экскурсия в музей искусств!
Понедельник
Предмет
Время
Математика
9:30
Русский язык
10:10
Информатика
12:00
Страница ученика, получивший уведомление
Авторизация
Имя пользователя
Пароль
ВОЙТИ
Страница авторизации


Класс
3В
06/06/2020
УЧИТЕЛЬ И.И.
Добавление предмета
Предмет
Мероприятие
Описание мероприятия
Экскурсия в музей искусств
Время
14:00
ОТМЕНА
СОЗДАТЬ
СОЗДАТЬ +
Предмет
Действия
Информатика
Добавление занятия
Класс
3В
06/01/2020
УЧИТЕЛЬ И.И.
Понедельник
СОЗДАТЬ +
Предмет
Время
Действия
Математика
9:30
Русский язык
10:10
Информатика
12:00
Страница учителя

Заключение

- Анализ предметной области;
- Рассмотрены технологии систем различных производителей;
- Рассмотрены существующие реализации подобных систем и был сделан анализ, в котором были выявлены достоинства и недостатки;
- Разработан алгоритм работы и был сделан выбор программного обеспечения для разработки приложения;.
- Разработана база данных, позволяющая хранить в себе массивы данных;
- Разработано программное обеспечение «Планировщик задач для инклюзивного образования».

Спасибо за внимание!