**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН**

**УПРАВЛЕНИЕ ОБРАЗОВАНИЯ ГОРОДА АЛМАТЫ**

**ИННОВАЦИОННЫЙ ТЕХНИЧЕСКИЙ КОЛЛЕДЖ ГОРОДА АЛМАТЫ**



**ОТЧЕТ**

**по учебной практике**

**Объектно-ориентированное программирование**

*(Название практики)*

**Назарханов Алишер**

*(Фамилия, имя, отчество)*

**Курс:** 3

**Группа:** П1 17-09Р

**Специальность:** 1304000 – “Вычислительная техника и программное обеспечение”

**Квалификация:** 1304043 – “техник-программист”

**Место прохождения практики:** ТОО «Инновационный технический колледж города Алматы»

*(наименование организации, адрес)*

**Период прохождения практики: с** « 17 **»**  апреля **20**  20 **г.**

**по** « 25 **»**  апреля **20**  20 **г.**

**Руководитель практики:** Карабалиев Ерлан

*(Фамилия, имя, отчество и должность)*

Подпись руководителя\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Проверил преподаватель:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Ф.И.О/

Защищен с оценкой «\_\_\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата защиты «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_\_г

СОДЕРЖАНИЕ

|  |  |  |
| --- | --- | --- |
|  | ВВЕДЕНИЕ | 3 |
| 1 | ОСНОВНАЯ ЧАСТЬ | 4 |
| 1.1 | Объектно-ориентированное программирование. ООП | 4 |
| 1.1.1 | Основные понятия и концепции ООП | 4 |
| 1.1.2 | Инкапсуляция | 5 |
| 1.1.3 | Наследование | 5 |
| 1.1.4 | Полиморфизм | 6 |
| 1.1.5 | Абстракции | 6 |
| 2 | СПЕЦИАЛЬНАЯ ЧАСТЬ | 8 |
| 2.1 | Постановка задачи | 8 |
| 2.2 | Этапы выполнения | 9 |
|  | ЗАКЛЮЧЕНИЕ | 22 |
|  | СПИСОК ЛИТЕРАТУРЫ | 23 |

ВВЕДЕНИЕ

Учебная практика по дисциплине «Объектно-ориентированное программирование» проходила с 17 апреля по 25 апреля в инновационном техническом колледже города Алматы и направлена на практическое закрепление, углубление и систематизацию знаний, полученных при изучении данной дисциплины.

Используя объектное мышление, можно легко проектировать сложные системы, оперируя объектами и взаимодействием между ними. Это значительно облегчает разработку программы, а так же дальнейшей её поддержке и расширении функционала.

ООП выполняет несколько важных задач:

* Бизнес процессы и сущности на уровне кода;
* Защита и получение только проверенных данных;
* Данные и функции объединены в одну сущность;

Основными задачами практики являются:

* Приобрети практической опыт использование методологии ООП;
* Приобретение и формирование умений практического опыта по дисциплине «Объектно-ориентированное программирование»;
* Изучить основные принципы построения связей между классами;
* Усовершенствовать навыки программирования на ЯП PHP;
* Получение знаний о работе Клиент-Серверного взаимодействия;

1. ОСНОВНАЯ ЧАСТЬ
   1. Объектно-ориентированное программирование. ООП

Объектно-ориентированное программирование (ООП) позволяет разложить проблему на составные части, каждая из которых становится самостоятельным объектом. Каждый из объектов содержит свой собственный код и данные, которые относятся к этому объекту.

Любая программа, написанная на языке ООП, отражает в своих данных состояние физических предметов либо абстрактных понятий – объектов программирования, для работы, с которыми она предназначена.

Все данные об объекте программирования и его связях с другими объектами можно объединить в одну структурированную переменную. В первом приближении ее можно назвать объектом.

С объектом связывается набор действий, иначе называемых методами. С точки зрения языка программирования набор действий или методов – это функции, получающие в качестве обязательного параметра указатель на объект и выполняющие определенные действия с данными объекта программирования. Технология ООП запрещает работать с объектом иначе, чем через методы, таким образом, внутренняя структура объекта скрыта от внешнего пользователя.

Описание множества однотипных объектов называется классом.

Объект – это структурированная переменная, содержащая всю информацию о некотором физическом предмете или реализуемом в программе понятии.

Класс – это описание множества объектов программирования (объектов) и выполняемых над ними действий.

Класс можно сравнить с чертежом, согласно которому создаются объекты. Обычно классы разрабатывают таким образом, чтобы их объекты соответствовали объектам предметной области решаемой задачи.

* + 1. Основные понятия и концепции ООП

Любая функция в программе представляет собой метод для объекта некоторого класса.

Класс должен формироваться в программе естественным образом, как только в ней возникает необходимость описания новых объектов программирования. Каждый новый шаг в разработке алгоритма должен представлять собой разработку нового класса на основе уже существующих.

Вся программа в таком виде представляет собой объект некоторого класса с единственным методом run (выполнить).

Программирование «от класса к классу» включает в себя ряд новых понятий.

Основными понятиями ООП являются:

* инкапсуляция;
* наследование;
* полиморфизм;
  + 1. Инкапсуляция

Инкапсуляция данных (от «капсула») – это механизм, который объединяет данные и код, манипулирующий с этими данными, а также защищает и то, и другое от внешнего вмешательства или неправильного использования. В ООП код и данные могут быть объединены вместе (в так называемый «черный ящик») при создании объекта.

Внутри объекта коды и данные могут быть закрытыми или открытыми. Закрытые коды или данные доступны только для других частей того же самого объекта и, соответственно, недоступны для тех частей программы, которые существуют вне объекта.

Открытые коды и данные, напротив, доступны для всех частей программы, в том числе и для других частей того же самого объекта.

По умолчанию, в классе (class) данные и методы приватные (private); они могут быть прочитаны и изменены только классом к которому принадлежат. Уровень доступа может быть изменен при помощи соответствующих ключевых слов которые предоставляет PHP.

В PHP доступно несколько спецификаторов, и они изменяют доступ к данным следующим образом:

* публичные (public) данные — доступны всем;
* защищенные (protected) — доступны только классу и дочерним классам;
* приватные (private) —доступны только классу которому они принадлежат.
  + 1. Наследование

Наследование. Новый, или производный класс может быть определен на основе уже имеющегося, или базового класса.

При этом новый класс сохраняет все свойства старого: данные объекта базового класса включаются в данные объекта производного, а методы базового класса могут быть вызваны для объекта производного класса, причем они будут выполняться над данными включенного в него объекта базового класса.

Иначе говоря, новый класс наследует как данные старого класса, так и методы их обработки.

Если объект наследует свои свойства от одного родителя, то говорят об одиночном наследовании. Если объект наследует данные и методы от нескольких базовых классов, то говорят о множественном наследовании.

Пример наследования – определение структуры, отдельный член которой является ранее определенной структурой.

В PHP для того чтобы унаследовать один класс от другого нужно после после указания названия класса написать extends и указать название того класса, от которого необходимо унаследоваться.

Пример: class Computer extends Device

* + 1. Полиморфизм

Полиморфизм – это свойство, которое позволяет один и тот же идентификатор (одно и то же имя) использовать для решения двух и более схожих, но технически разных задач.

Целью полиморфизма, применительно к ООП, является использование одного имени для задания действий, общих для ряда классов объектов. Такой полиморфизм основывается на возможности включения в данные объекта также и информации о методах их обработки (в виде указателей на функции).

Будучи доступным в некоторой точке программы, объект , даже при отсутствии полной информации о его типе, всегда может корректно вызвать свойственные ему методы.

Полиморфная функция – это семейство функций с одним и тем же именем, но выполняющие различные действия в зависимости от условий вызова.

* + 1. Абстракции

Абстракция — в объектно-ориентированном программировании это придание объекту характеристик, которые отличают его от всех других объектов, четко определяя его концептуальные границы. Основная идея состоит в том, чтобы отделить способ использования составных объектов данных от деталей их реализации в виде более простых объектов, подобно тому, как функциональная абстракция разделяет способ использования функции и деталей её реализации в терминах более примитивных функций, таким образом, данные обрабатываются функцией высокого уровня с помощью вызова функций низкого уровня.

Такой подход является основой объектно-ориентированного программирования. Это позволяет работать с объектами, не вдаваясь в особенности их реализации. В каждом конкретном случае применяется тот или иной подход: инкапсуляция, полиморфизм или наследование. Например, при необходимости обратиться к скрытым данным объекта, следует воспользоваться инкапсуляцией, создав, так называемую, функцию доступа или свойство.

1. СПЕЦИАЛЬНАЯ ЧАСТЬ
   1. Постановка задачи

Разработать веб сайт для просмотра расписание занятий. Просматривать своё расписание будут как студенты разных курсов, так и преподаватели.

Также необходимо разработать страницы для администрования сайта т.е добавления новых групп и преподавателей, удаления ненужного расписания, изменения неактуальной информации о расписании.

На данном сайте будут существовать две роли: гость и админ. Гостем будут являться неавторизованные пользователи, которые хотят лишь просмотреть своё расписание т.е это студенты и преподаватели. Админом будет являться главный администратор, который управляет контентом сайта и может добавлять, удалять, изменять информацию о группах, преподавателях и расписании.

Используемые технологии: HTML, CSS, SCSS, JavaScript, PHP.

Используемые инструменты: VS Code, Live Server.

* 1. Этапы выполнения

1. Первым делом начну со структуры проекта. В корневой директории проекта находится index.php файл, который является точкой входа в проект. В нём и описан веб сервер (Backend). Запустить его можно при помощи Apache сервера.

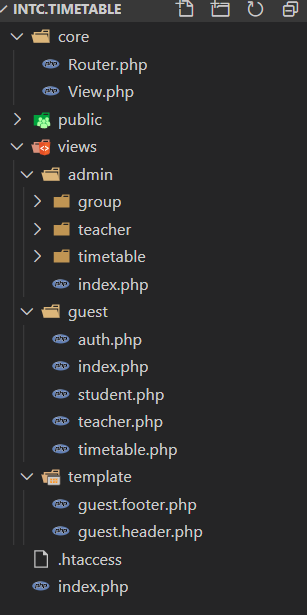


Рис. 2.1 Структура проекта

1. Также в корне проекта существуют две папки public и views. В папке public находятся все статические файлы относящиеся к фронтенду проекта т.е это css и js исходники. Кроме этого, css файлы написаны в формате SCSS, что позволяет объединить все css файлы в один и подключать этот файл в html единожды.
2. В папке views находятся html шаблоны сайта, которые шаблонизируются при помощи php. Папка также разделена на два раздела: guest и admin. Так пользователи увидят страницы соответствующие только их роли.

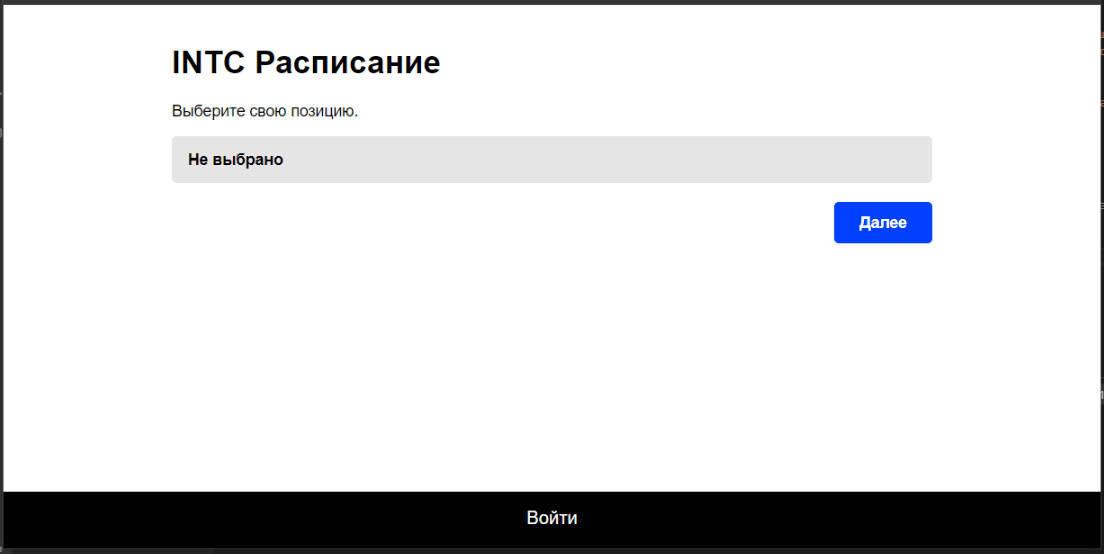


Рис. 2.2 Главная страница сайта

1. Когда пользователь переходит на главную страницу, он должен выбрать свою позицию (студент или преподаватель). Разметка этой страницы выглядит следующим образом:

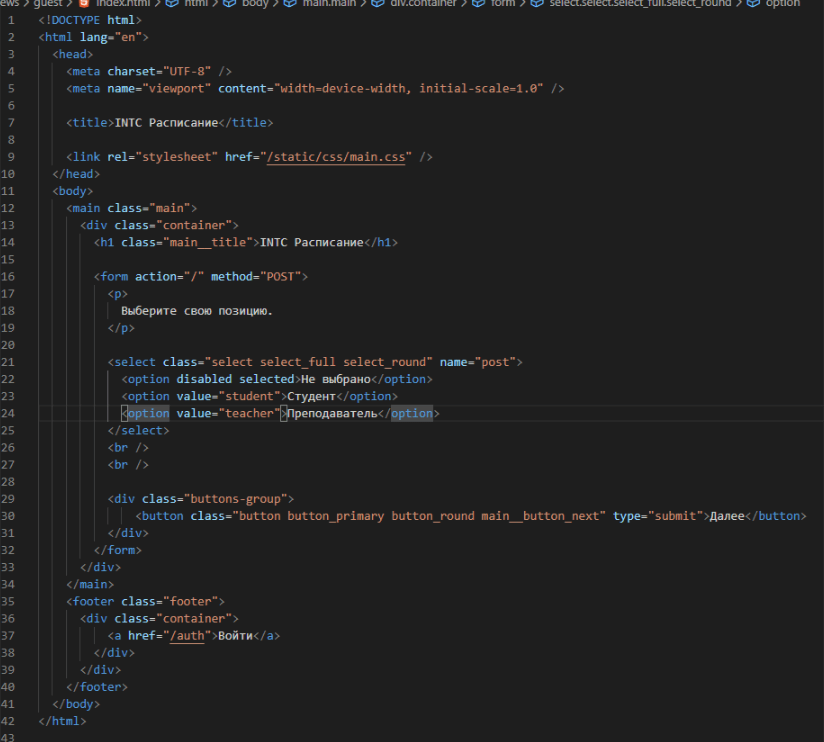


Рис. 2.3 HTML разметка главной страницы

1. Страница поделена на две секции: main и footer. Секции обернуты в соответствующие семантические тэги. Внутри main находится форма, которая отправляет данные формы на сервер т.е пост того, кто зашел на сайт.

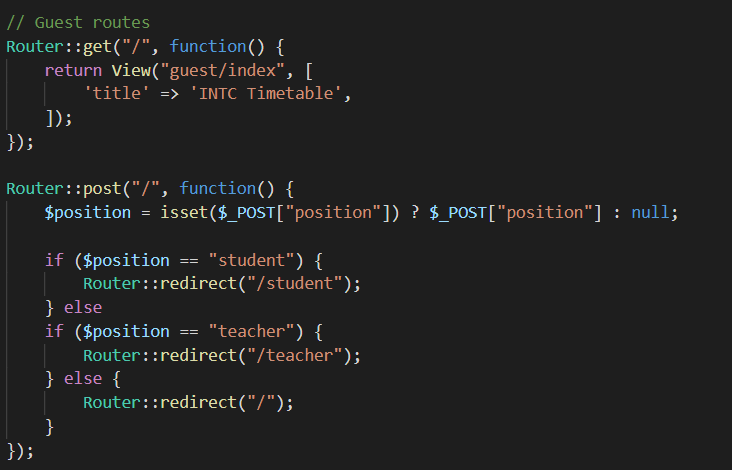


Рис. 2.4 PHP код, обрабатывающий главный маршрут

1. После того, как форма была отправлена на сервере будет вызвана функция с помощью класса Router, которая перенаправляет пользователя на соответствующую страницу (/student или /teacher).

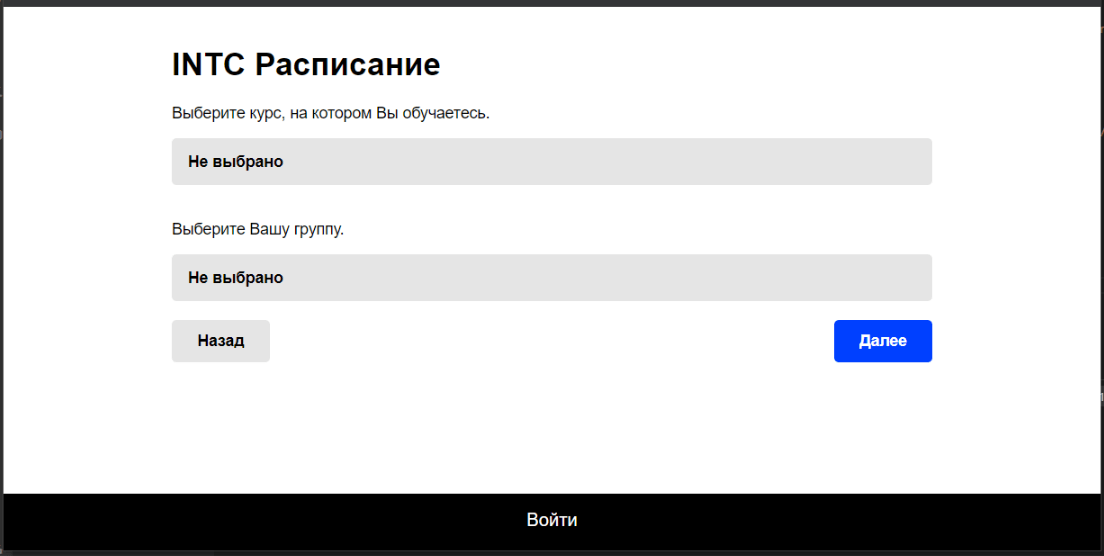


Рис. 2.5 Страница для студентов (/student)

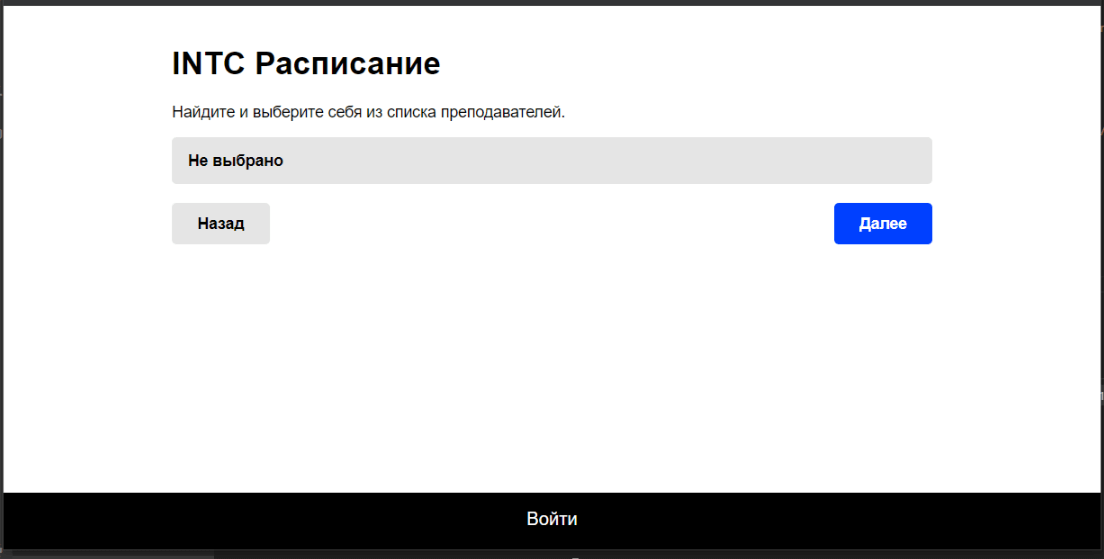


Рис. 2.6 Страница для преподавателей (/teacher)

1. Студент для просмотра своего расписания должен сначала выбрать свой курс, а после выбрать свою группу и перейти "Далее". В то время как преподаватель долэен лишь выбрать себя из списка преподавателей. HTML разметка данных страниц лишь немного отличается от главной.

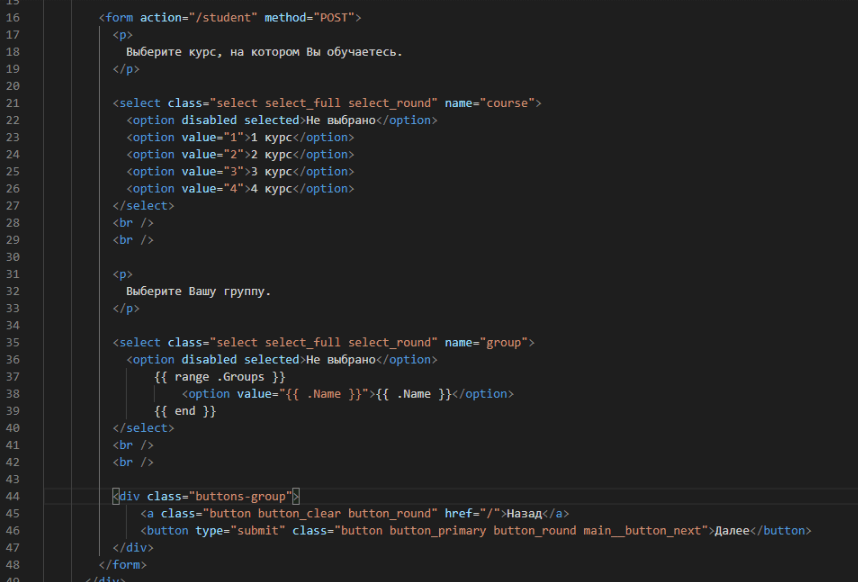


Рис. 2.7 HTML разметка страницы студента

1. Для получения страницы студента, запускается функция Router::get(/student).

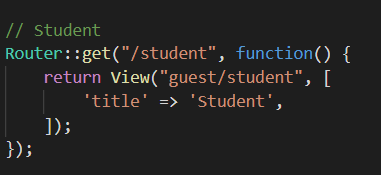


Рис. 2.8 PHP код, обрабатывающий роут студента

1. После отправки формы, студент получает информацию о его расписании занятий.

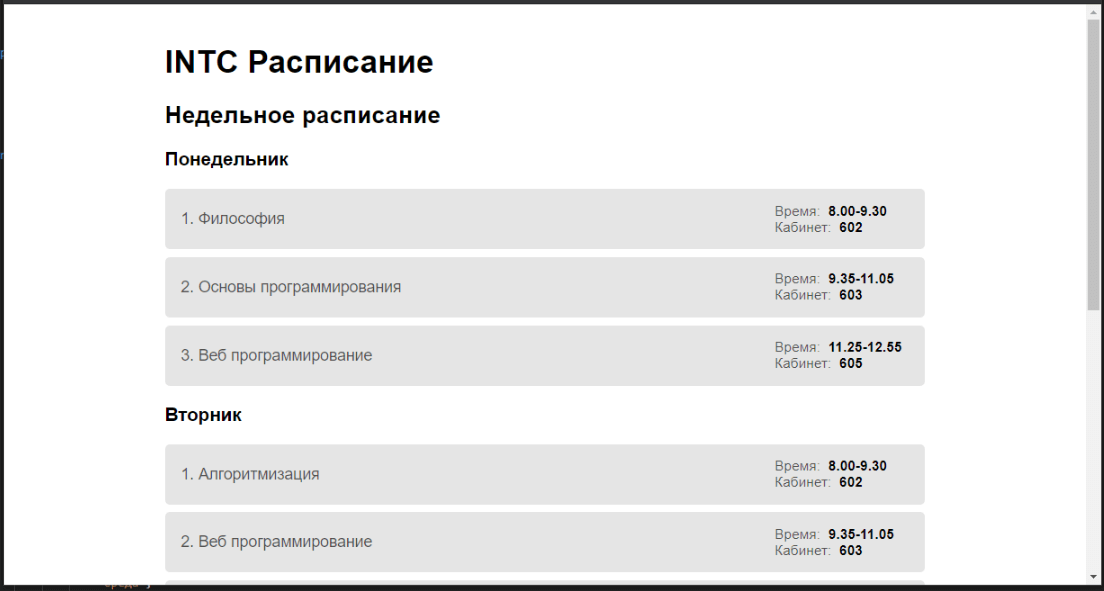


Рис. 2.9 Страница с расписанием для студентов (/timetable)

1. На HTML разметке страниц не раз уже была написана конструкция <?php echo ...; ?>. В эти места PHP интерпретатор вставляет, которые мы передаем в шаблон.

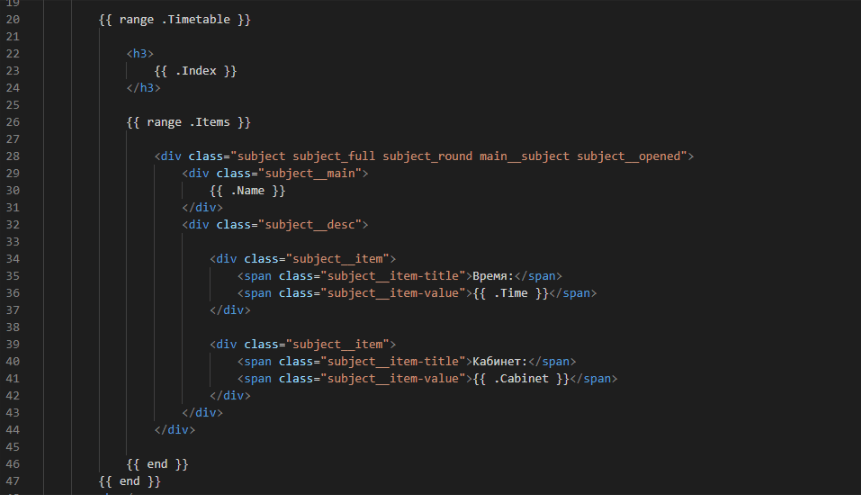


Рис. 2.10 HTML разметка страницы расписания

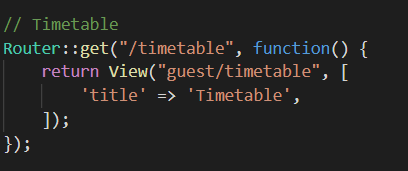


Рис. 2.11 PHP код, обрабатывающий роут расписания

1. Страница авторизовации реализована только для входа в панель администратора. Для входа необходимо заполнить два обязательных поля: имя пользователя и пароль.

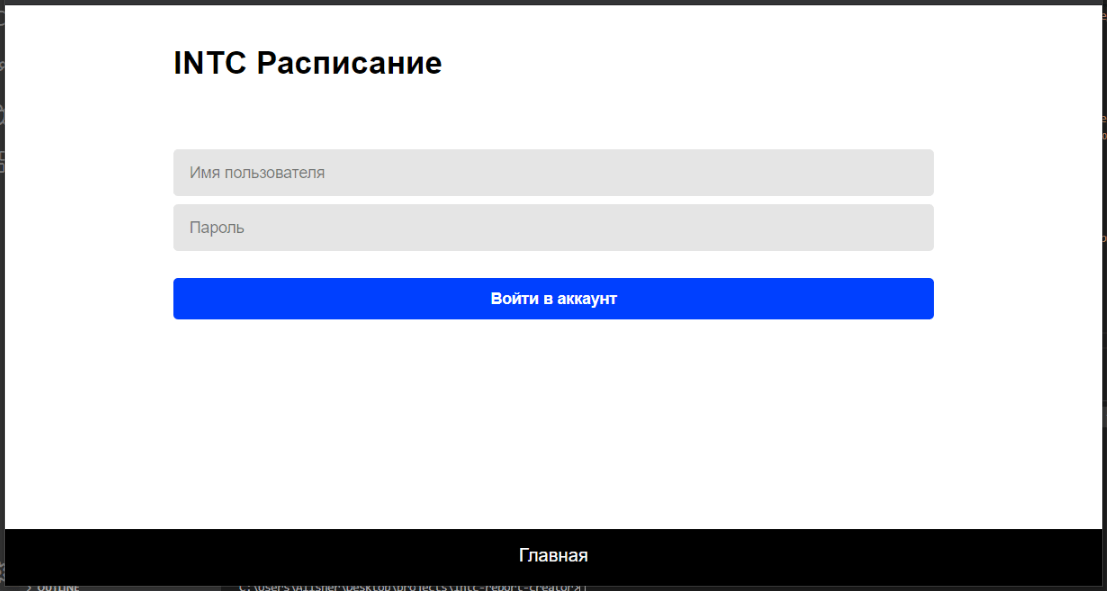


Рис. 2.12 Страница с входа в админ панель (/auth)

1. Разметка страницы состоит из двух input элементов, которые обернуты в элемент form. При нажатии элемент button форма отправляется и происходит авторизация.

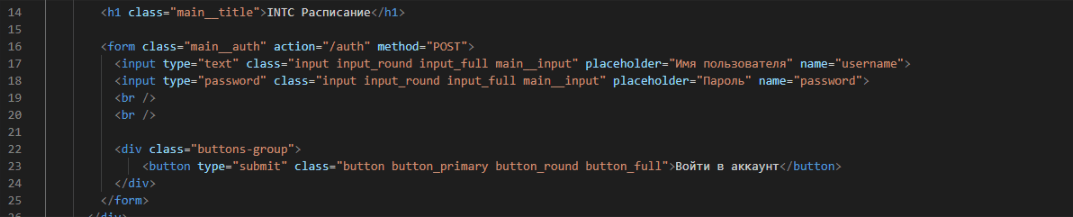


Рис. 2.13 HTML разметка страницы авторизации (/auth)

1. После входа, админ попадает на страницу панели управления сайта. Страница визуально поделена на три части: меню, шапка и основная часть. В меню страницы можно полностью управлять информацией сайта.

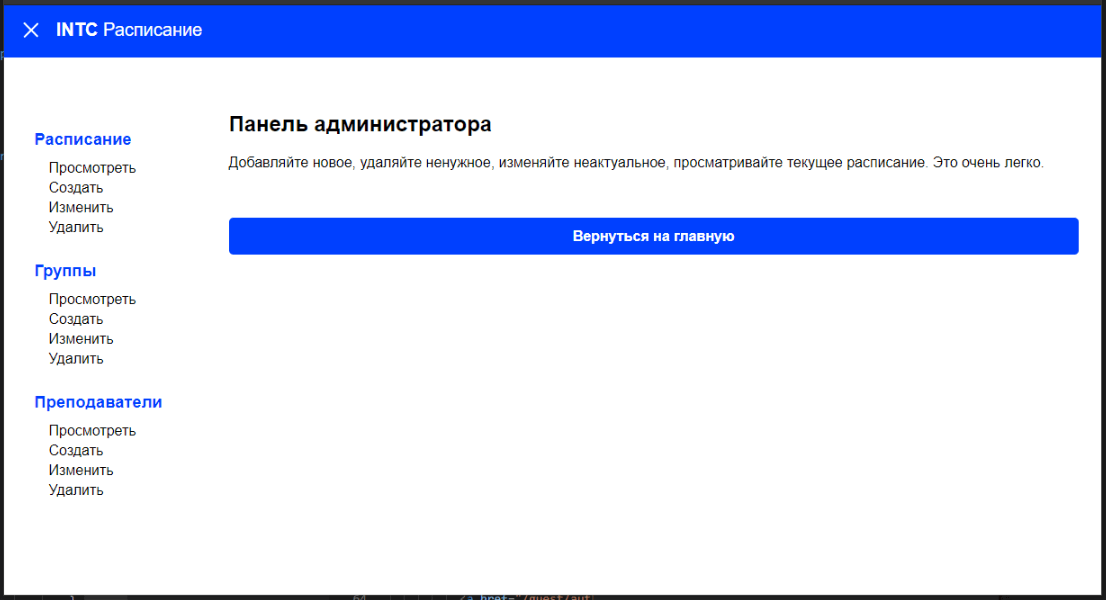


Рис. 2.14 Страница админ панели (/admin)

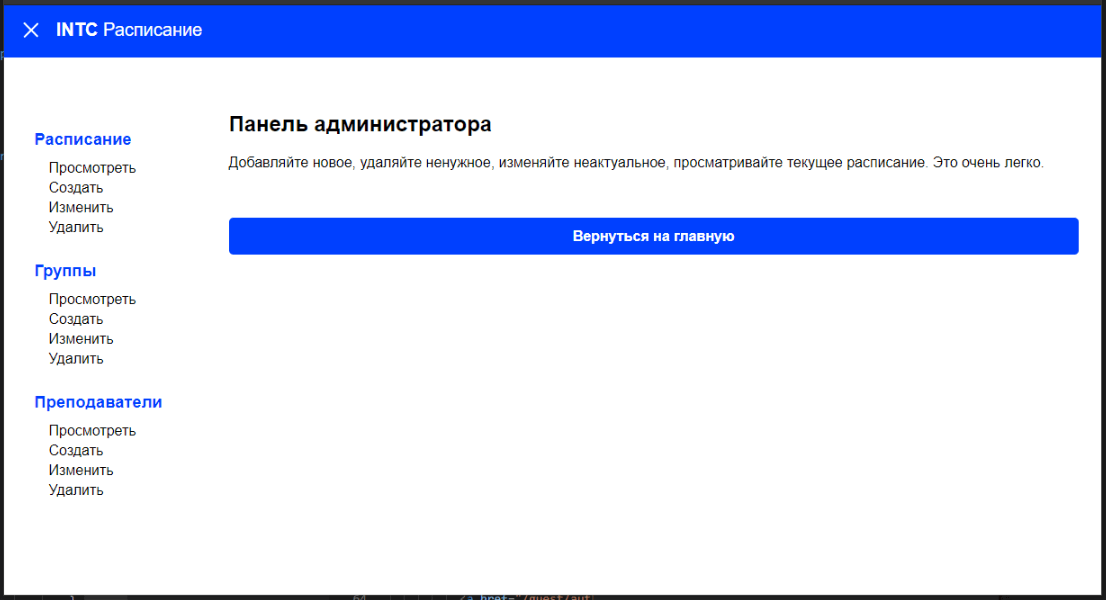


Рис. 2.15 HTML разметка админ панели (/admin)

1. Стили данной панели администратора отличаются от обычного вида сайта для "гостей". В стилях используются новые вохможности CSS Variables, позволяющие быстро менять палитру цветов на сайте.

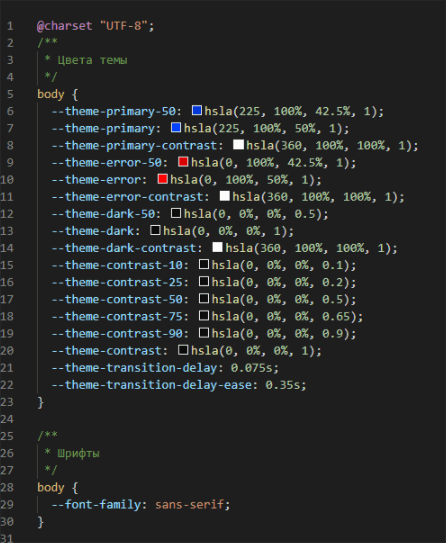


Рис. 2.16 CSS стили админ панели (/admin)

1. Перейдя на страницу "Просмотр групп", мы увидим список существующих групп на сайте. Справа от их названия есть кнопка с карандашем, позволяющая изменить информацию об этой группе.

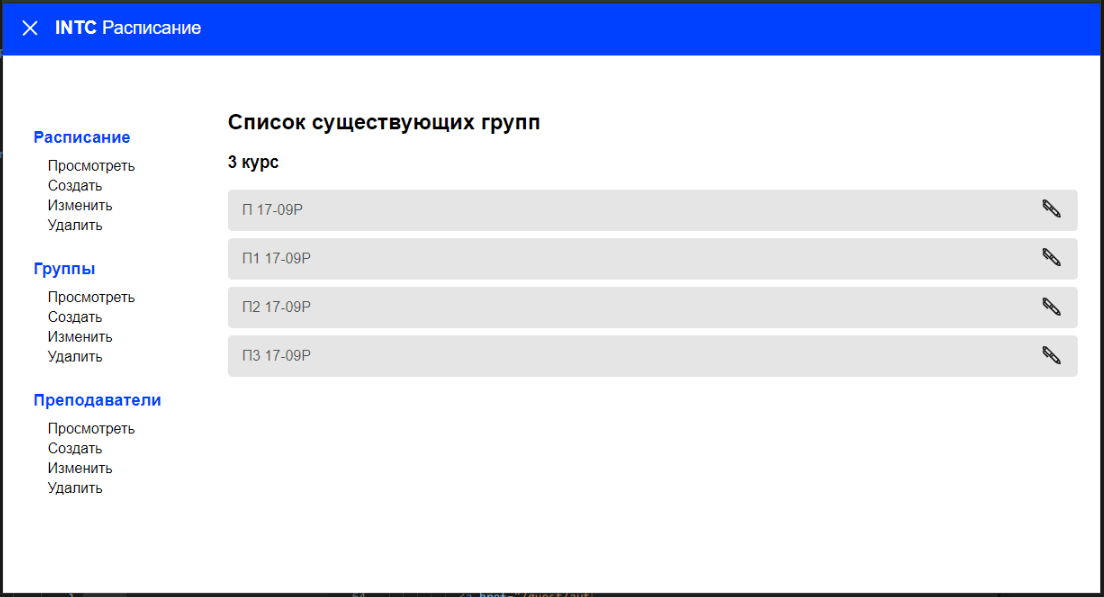


Рис. 2.17 Страница просмотра групп (/admin/groups)

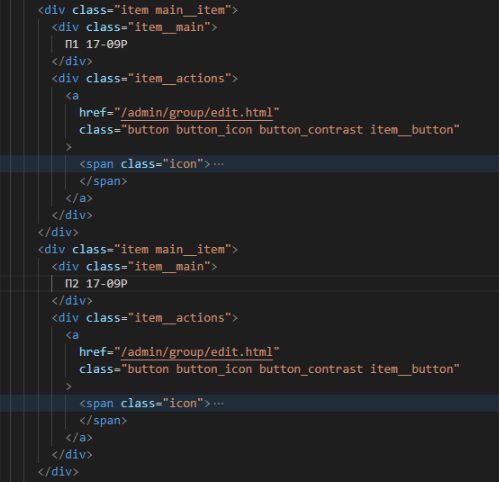


Рис. 2.18 HTML разметка страницы просмотра групп (/admin/groups)

1. Стили ниже описывают как будет выглядеть стандартное разделение сайта на секции. Также там мы обнуляем значения установленные браузером по умолчанию.

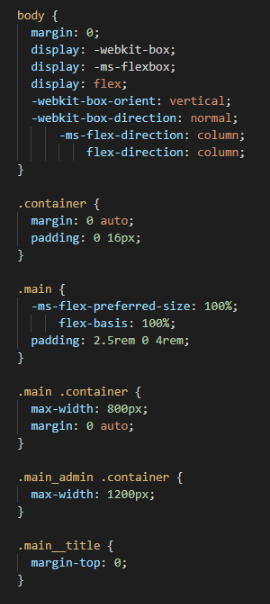


Рис. 2.19 CSS стили админ панели (/admin)

1. Перейдя на страницу "Создание групп", мы увидим поля "Курс группы" и "Название группы". Их нужно заполнить и после нажатия кнопки добавить, данная группа будет добавлена на сайт.

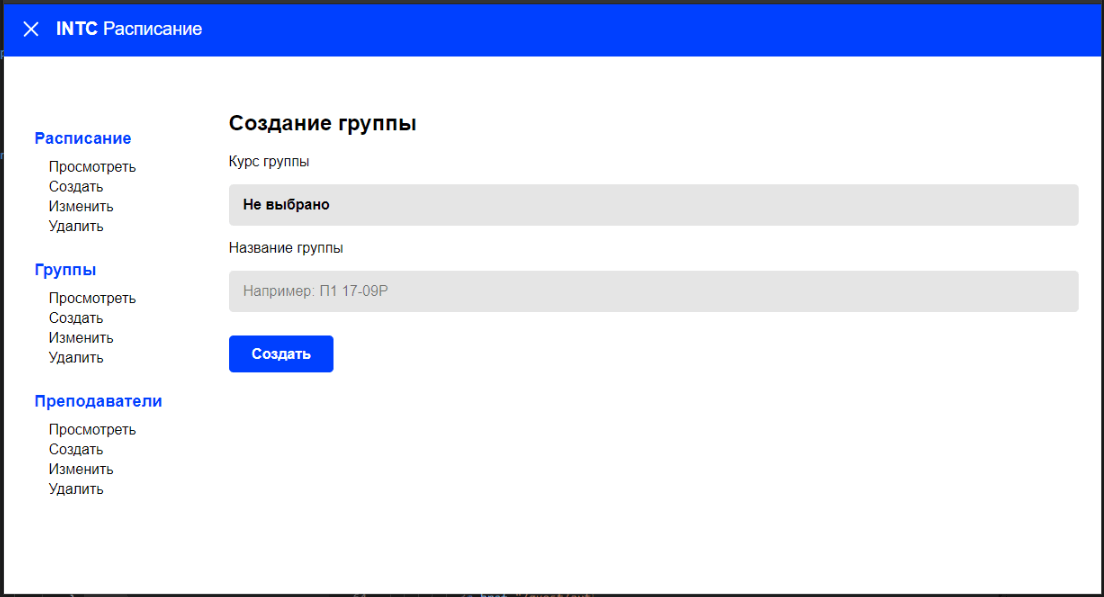


Рис. 2.20 Страница создания группы (/admin/groups/create)

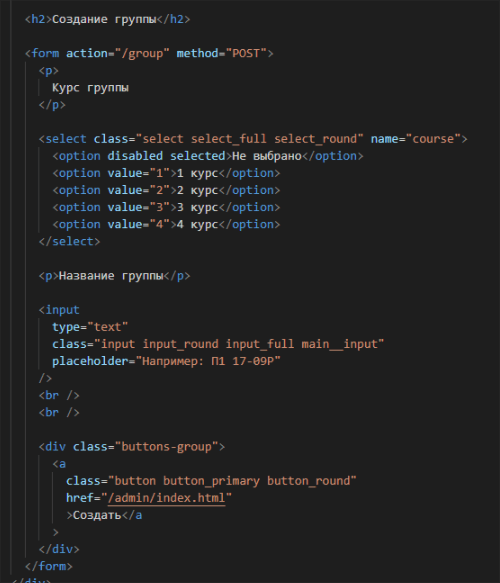


Рис. 2.21 HTML разметка страницы создания групп (/admin/groups/create)

1. Стили ниже описывают как будет выглядеть меню навигации по сайту. Из стилей видно что с помощью блока @keyframes мы описали нашу анимацию и применили её к меню с помощью свойства animation.

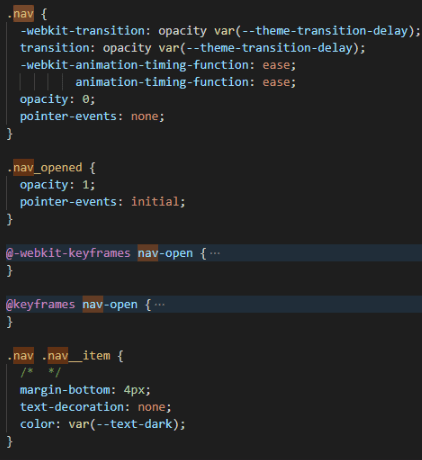


Рис. 2.22 CSS стили админ панели (/admin)

1. Перейдя на страницу "Изменения групп", мы увидим поля "Курс группы" и "Название группы". Их нужно изменить на те значения, которые нам необходимы и после нажатия кнопки изменить, инфа о данной группе будет изменена на сайте.

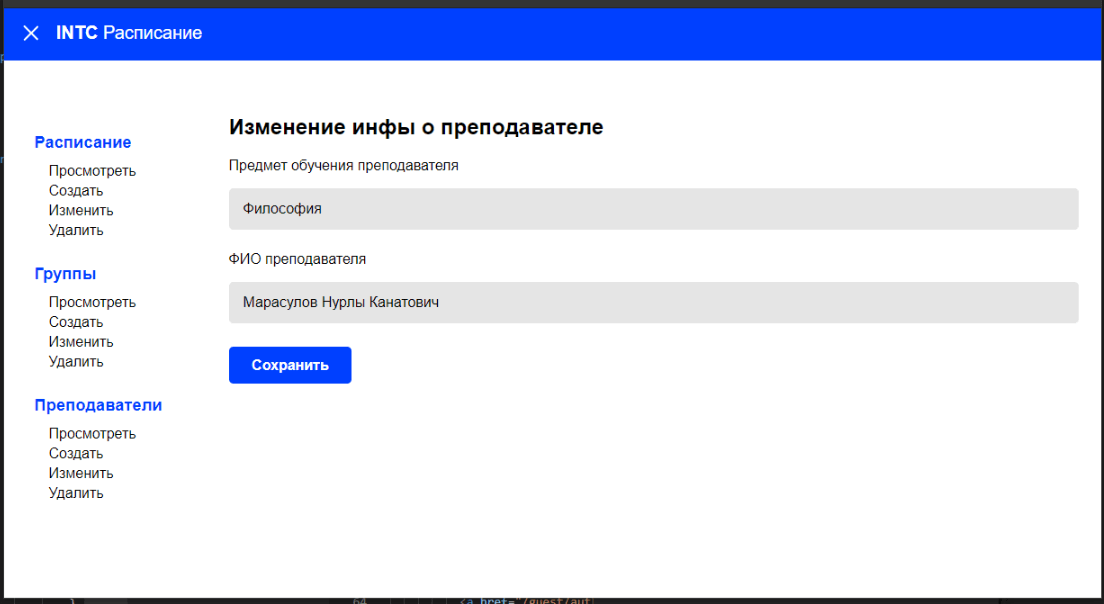


Рис. 2.23 Страница создания группы (/admin/groups/edit)

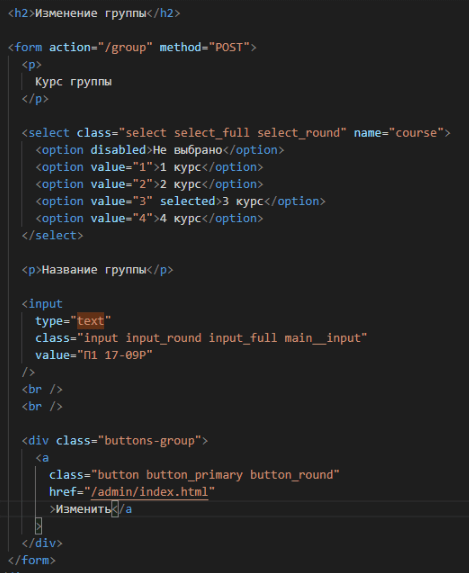


Рис. 2.24 HTML разметка страницы создания групп (/admin/groups/edit)

1. Стили ниже описывают как будет выглядеть синяя кнопка, которая используется по всему сайту.

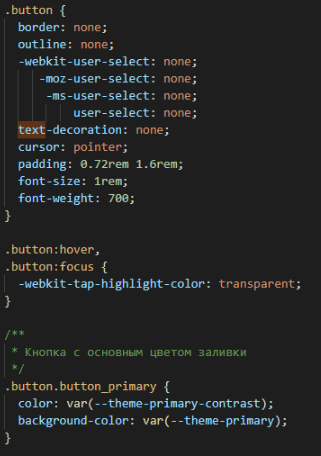


Рис. 2.25 CSS стили админ панели (/admin)

1. Перейдя на страницу "Изменения групп", мы увидим поля "Курс группы" и "Название группы". Их нужно изменить на те значения, которые нам необходимы и после нажатия кнопки изменить, инфа о данной группе будет изменена на сайте.

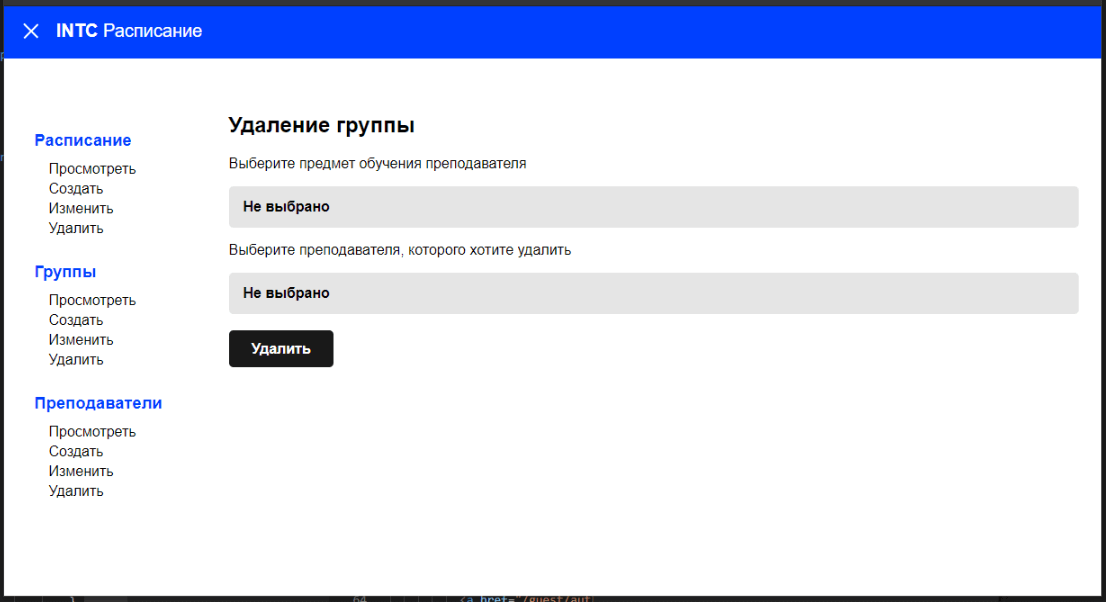


Рис. 2.26 Страница создания группы (/admin/groups/delete)

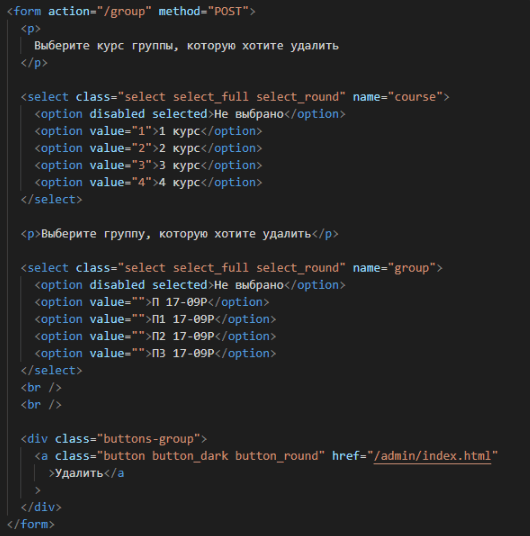


Рис. 2.27 HTML разметка страницы создания групп (/admin/groups/delete)

1. Стили ниже описывают как будет выглядеть поле ввода информации, которая используется по всему сайту.

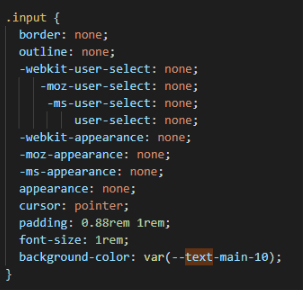


Рис. 2.28 CSS стили админ панели (/admin)

1. Страницы добавления, изменения и удаления преподавателей выглядят аналогичным образом. Таким образом был создан сайт с просмотром расписаний занятий удобный для каждого.

ЗАКЛЮЧЕНИЕ

Практика была пройдена по дисциплине «Объектно-ориентированное программирование». За время практики я изучил ту самую базу работы c разработкой, необходимую для работы над реальными задачами. Проанализировал много новой для себя информации, на практике применил знания и умения. Практика помогла закрепить навыки, полученные во время обучения. Преподаватель дал мне массу ценных советов. Я приобрел незаменимый опыт, который, несомненно, очень пригодится мне в дальнейшей профессиональной деятельности. На мой взгяд, разработка не так сложна как казалось ранее и каждый из нас может понять её, решая реальные бизнес задачи.

Результатом прохождения практики стало:

* Подробное ознакомление с разработкой по методологии ООП;
* Получение практических навыков и умений в разработке;
* Изучены основы HTML, JS, CSS, PHP;
* Усвоены материал и практические задания;
* Получены бесценные, профессиональные, практические навыки;

СПИСОК ЛИТЕРАТУРЫ

1. Выразительный JavaScript. Современное веб-программирование М. Хавербеке (Издательство Питер, 2019 г.)
2. Изучаем JavaScript: руководство по созданию современных веб-сайтов. 3-е издание Э. Браун (Издательство Диалектика Альфа-книга, 2017 г.
3. HTML5. Карманный справочник Д. Роббинс (Издательство Диалектика / Вильямс Вильямс, 2019 г.
4. CSS. Полный справочник. Визуальное форматирование веб-страниц Э. Уэйл, Мейер Э. А. (Издательство Диалектика Диалектика / Вильямс, 2019 г.
5. JavaScript и jQuery. Исчерпывающее руководство Макфарланд Д.С. (Издательство Эксмо, 2015 г.
6. HTML, JavaScript, PHP и MySQL. Джентльменский набор Web-мастера / Н.А. Прохоренок, В.А. Дронов. - Москва: СПб. [и др.] : Питер, 2015. - 768 c.
7. PHP и MySQL; НТ Пресс - М., 2016. - 569 c. Гизберт Дамашке
8. Разработка современных веб-приложений. Анализ предметных областей и технологий Эспозито Д. (Издательство Вильямс Альфа-книга, 2017 г.
9. AJAX. Программирование для Интернета Бенкен Е.С. (Издательств БХВ-Петербург, 2009 г.
10. Проектирование веб-интерфейсов Б. Скотт, Нейл Т. (Издательство Символ-Плюс Символ, 2018 г.
11. Как устроен JavaScript Д. Крокфорд (Издательство Питер, 2019 г.
12. JavaScript для профессиональных веб-разработчиков — Николас Закас (Издательство: Питер, 2015 г.