|  |  |
| --- | --- |
| **RTK_LOGO.jpg** | **Санкт-Петербургское Государственное бюджетное профессиональное образовательное учреждение «Радиотехнический колледж»**  199155, Санкт-Петербург, наб. реки Смоленки, д.1  Тел.: (812) 405-85-59 факс (812) 405-85-59 http://www.rtplspb.ru, e-mail: info@spb-rtk.ru |

**Отчет**

об учебной практике

по профессиональному модулю

ПМ.05: «Проектирование и разработка информационных систем»

по программе подготовки специалистов среднего звена, специальности 09.02.07. Информационные системы и программирование

4 семестр

сроки прохождения практики

с «2» мая 2023 по «2» июня 2023

\_Шабанова Виктория Юрьевна\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Ф.И.О.)

Руководитель практики: Харин Евгений Николаевич

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 6](#_Toc136956228)

[1 Создание use-case диаграммы 7](#_Toc136956229)

[2 Создание IDEF0 диаграммы нулевого уровня 9](#_Toc136956230)

[3 Декомпозиция IDEF0 диаграммы 1 уровня 11](#_Toc136956231)

[3.1 Блок «Работа с пользователем» 11](#_Toc136956232)

[3.2 Блок «Работа с постами» 12](#_Toc136956233)

[3.3 Блок «Работа с комментариями» 13](#_Toc136956234)

[3.4 Блок «Работа администратора» 14](#_Toc136956235)

[4 Декомпозиция IDEF0 диаграммы 2 уровня в нотации DFD 15](#_Toc136956236)

[4.1 Декомпозиция блока «Работа с пользователем» 15](#_Toc136956237)

[4.2 Декомпозиция блока «Работа с постами» 16](#_Toc136956238)

[5 Создание ER диаграммы базы данных 17](#_Toc136956239)

[6 Создание диаграммы компонентов системы 19](#_Toc136956240)

[7 Реализация информационной системы 21](#_Toc136956241)

[7.1 Подготовка к реализации функций информационной системы 21](#_Toc136956242)

[7.2 Создание класса Request 22](#_Toc136956243)

[7.2.1 Конструктор 22](#_Toc136956244)

[7.2.2 Метод clearParameter 22](#_Toc136956245)

[7.2.3 Метод clearArray 23](#_Toc136956246)

[7.2.4 Метод post 23](#_Toc136956247)

[7.2.5 Метод get 24](#_Toc136956248)

[7.2.6 Метод getHost 24](#_Toc136956249)

[7.3 Создание класса MySql 24](#_Toc136956250)

[7.3.1 Конструктор 25](#_Toc136956251)

[7.3.2 Метод myQuery 25](#_Toc136956252)

[7.3.3 Метод uniqueCheck 25](#_Toc136956253)

[7.4 Создание класса Data 26](#_Toc136956254)

[7.4.1 Метод validateData 26](#_Toc136956255)

[7.4.2 Метод loadData 26](#_Toc136956256)

[7.4.3 Метод rnReplace 27](#_Toc136956257)

[7.4.4 Метод brReplace 27](#_Toc136956258)

[7.4.5 Метод changeDate 28](#_Toc136956259)

[7.4.6 Метод sqlFormat 28](#_Toc136956260)

[7.5 Создание класса User 28](#_Toc136956261)

[7.5.1 Конструктор 29](#_Toc136956262)

[7.5.2 Метод isAdmin 30](#_Toc136956263)

[7.5.3 Метод load 30](#_Toc136956264)

[7.5.4 Метод validateRegister 30](#_Toc136956265)

[7.5.5 Метод save 31](#_Toc136956266)

[7.5.6 Метод validateLogin 32](#_Toc136956267)

[7.5.7 Метод login 32](#_Toc136956268)

[7.5.8 Метод identity 33](#_Toc136956269)

[7.5.8 Метод logout 34](#_Toc136956270)

[7.5.9 Метод unblock 34](#_Toc136956271)

[7.6 Создание класса Response 35](#_Toc136956272)

[7.6.1 Конструктор 35](#_Toc136956273)

[7.6.2 Метод getLink 35](#_Toc136956274)

[7.6.3 Метод redirect 36](#_Toc136956275)

[7.7 Реализация регистрации пользователя 37](#_Toc136956276)

[7.8 Реализация авторизации пользователя 38](#_Toc136956277)

[7.9 Создание меню информационной системы 39](#_Toc136956278)

[7.9.1 Конструктор 39](#_Toc136956279)

[7.9.2 Метод appear 40](#_Toc136956280)

[7.10 Создание класса Post 40](#_Toc136956281)

[7.10.1 Метод validate 41](#_Toc136956282)

[7.10.2 Метод load 42](#_Toc136956283)

[7.10.3 Метод save 42](#_Toc136956284)

[7.10.4 Метод findOne 43](#_Toc136956285)

[7.10.5 Метод dateAppear 43](#_Toc136956286)

[7.10.6 Метод postsList 44](#_Toc136956287)

[7.10.7 Метод list10 44](#_Toc136956288)

[7.10.8 Метод delete 45](#_Toc136956289)

[7.11 Реализация функции создания и редактирования поста 45](#_Toc136956290)

[7.12 Отображения постов на главной странице 47](#_Toc136956291)

[7.13 Отображение списка постов на странице со всеми постами 48](#_Toc136956292)

[7.14 Создание класса Comment 50](#_Toc136956293)

[7.14.1 Метод validate 51](#_Toc136956294)

[7.14.2 Метод load 51](#_Toc136956295)

[7.14.3 Метод save 52](#_Toc136956296)

[7.14.4 Метод dateAppear 52](#_Toc136956297)

[7.14.5 Метод commentList 52](#_Toc136956298)

[7.14.6 Метод delete 53](#_Toc136956299)

[7.15 Создание класса Admin 53](#_Toc136956300)

[7.15.1 Метод block 54](#_Toc136956301)

[7.15.2 Метод commentDelete 54](#_Toc136956302)

[7.15.3 Метод postDelete 54](#_Toc136956303)

[7.15.4 Метод userPostsDelete 55](#_Toc136956304)

[7.16 Отображение поста на странице просмотра поста 56](#_Toc136956305)

[7.17 Реализация функции удаления поста 57](#_Toc136956306)

[7.18 Реализация функций работы с комментариями 58](#_Toc136956307)

[7.19 Реализация отображения списка пользователей 61](#_Toc136956308)

[7.20 Реализация постоянной блокировки пользователей 62](#_Toc136956309)

[7.21 Реализация временной блокировки пользователя 63](#_Toc136956310)

[8 Составление Mind Map системы 64](#_Toc136956311)

[9 Составление тест-кейсов для тестирования интерфейса системы 65](#_Toc136956312)

[ЗАКЛЮЧЕНИЕ 69](#_Toc136956313)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 70](#_Toc136956314)

ВВЕДЕНИЕ

Специальность 09.02.07. Информационные системы и программирование, является очень востребованной в настоящее время, поскольку, в связи с активным развитием прогресса в области информационных технологий, этой сфере продолжают требоваться новые сотрудники и исследователи.

Целью данного отчета будет практико-теоретическое воспроизведение содержания учебной практики. Вся оценочная документация, отраженная в дневнике по практике и характеристике будет предложена руководителю практики от ОУ в виде отдельных документов.

В качестве средств документирования практико-теоретических результатов УП будут использоваться скриншоты и фотографии с пояснениями различных этапов выполнения работ. Поскольку для выполнения заданий необходимо уделять внимание теоретической составляющей — формированию основных понятий и представлений, связанных с предметной областью, в отчет будет включена информация из дополнительных источников. Результаты работы будут ежедневно фиксироваться в дневнике по практике и сопровождаться оценкой руководителя практики, в отчете эти данные представлены не будут.

1 Создание use-case диаграммы

В ходе проектирования информационной системы была составлена диаграмма вариантов использования, которая показывает роли пользователей и их действия. Она представлена на рисунке 1.

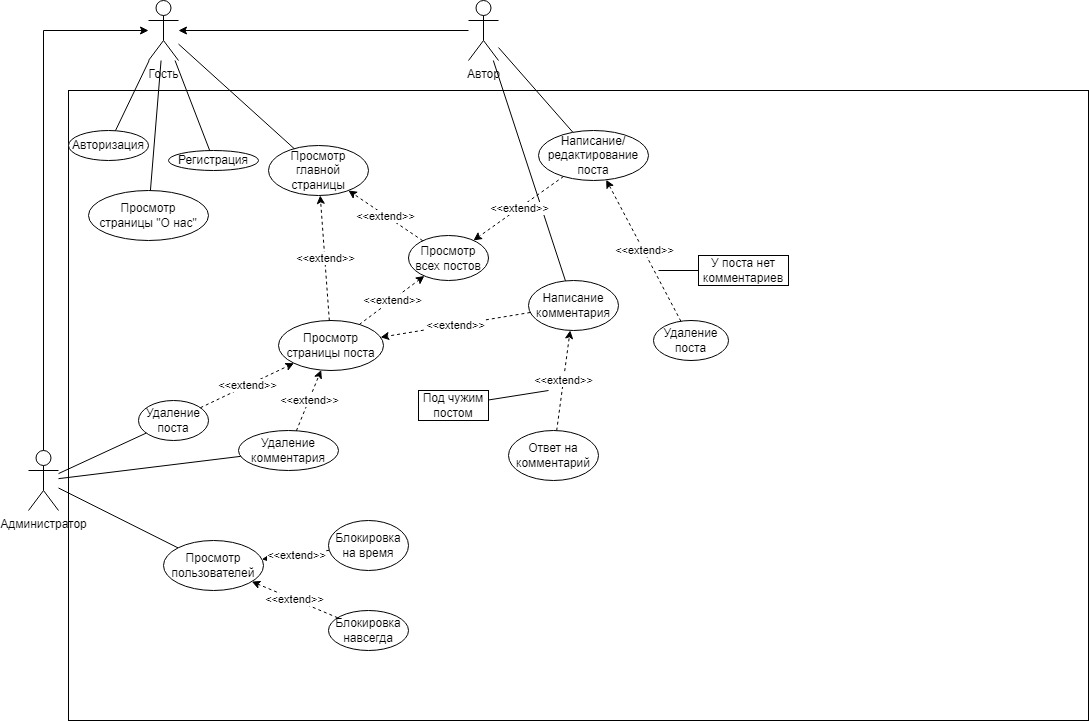


Рисунок 1 - Use-case диаграмма

Выделенные актёры:

- гость – не авторизованный пользователь;

- автор – авторизованный пользователь;

- администратор – авторизованный пользователь с правами управления системой.

Выделенные прецеденты для актёров:

- регистрация – позволяет гость зарегистрироваться;

- авторизация – позволяет гостю, который уже зарегистрировался, пройти авторизацию;

- просмотр страницы «О нас» - позволяет гостю, автору и администратору посещать страницу сайта «О нас»;

- просмотр главной страницы – позволяет гостю, автору и администратору посещать главную страницу сайта, на которой содержится лента превью 10 последних добавленных постов;

- просмотр всех постов – позволяет гостю, автору и администратору просматривать превью всех постов, которые есть на сайте;

- просмотр главной страницы поста – позволяет гостю, автору и администратору видеть текст поста и комментарии к нему;

- написание комментария – позволяет автору добавить свой комментарий к чужому посту;

- ответ на комментарий – позволяет автору ответить на чьей-то комментарий к любому посту;

- написание/редактирование поста – позволяет автору создать новый или изменить старый пост;

- удаление поста – позволяет автору удалить свой пост, если к нему не было оставлено комментариев, позволяет администратору удалить пост;

- удаление комментария – позволяет администратору удалить чей-то комментарий к посту;

- просмотр пользователей - позволяет администратору смотреть список всех пользователей сайта;

- блокировка на время – позволяет администратору заблокировать на определенное время пользователя из списка;

- блокировка навсегда – позволяет администратору заблокировать навсегда любого пользователя из списка.

2 Создание IDEF0 диаграммы нулевого уровня

При проектировании информационной системы необходимо провести структурный анализ, предметом которого будут её функции. Для этого составляются диаграммы в нотации IDEF0. У IDEF0 диаграммы 0 уровня есть один функциональный блок «Блог по книге Вадима Панова “Тайный город”». Сама диаграмма представлена на рисунке 2.

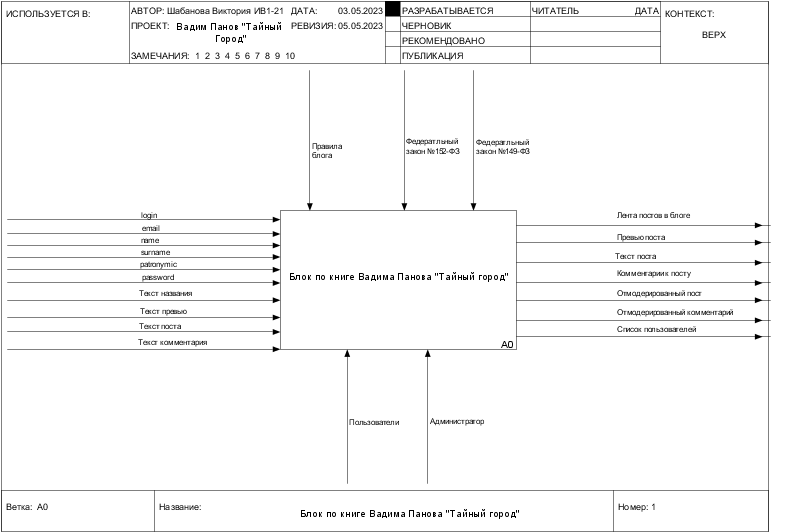


Рисунок 2 – IDEF0 диаграмма 0 уровня

Входящие дуги:

-login,

-email,

-name,

-surname,

-patronymic

-password,

-текст названия,

-текст превью,

-текст поста,

-текст комментария.

Дуги механизма:

-пользователи,

-адмнинистратор.

Дуги управления:

-правила блога,

-федеральный закон №152-ФЗ,

-федеральный закон №149-ФЗ.

Выходящие дуги:

-лента постов в блоге,

-превью поста,

-текст поста,

-комментарии к посту,

-отмодерированный пост,

-отмодерированный комментарий,

-список пользователей.

3 Декомпозиция IDEF0 диаграммы 1 уровня

Для упрощения анализа и разработки информационной системы осуществляется декомпозиция. Декомпозиция функционального блока IDEF0 диаграммы нулевого уровня представлена на рисунке 3. В ней содержится четыре функциональных блока:

-работа с пользователем,

-работа с постами,

-работа с комментариями,

-работа администратора.

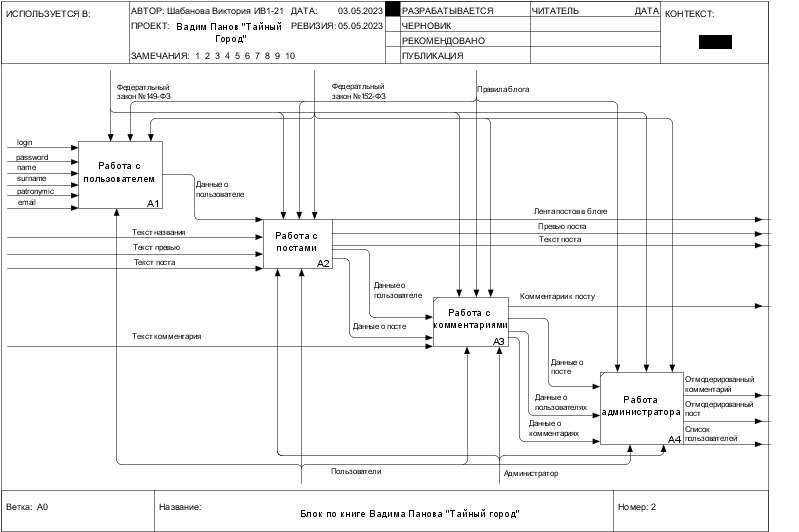
****

Рисунок 3 - IDEF0 диаграмма 1 уровня

3.1 Блок «Работа с пользователем»

Входящие дуги:

-login,

-email,

-name,

-surname,

-patronymic,

-password.

Дуги механизма:

-пользователи.

Дуги управления:

-правила блога,

-федеральный закон №152-ФЗ,

-федеральный закон №149-ФЗ.

Выходящие дуги:

-данные о пользователе.

3.2 Блок «Работа с постами»

Входящие дуги:

-данные о пользователе,

-текст названия,

-текст превью,

-текст поста.

Дуги механизма:

-пользователи,

-адмнинистратор.

Дуги управления:

-правила блога,

-федеральный закон №152-ФЗ,

-федеральный закон №149-ФЗ.

Выходящие дуги:

-данные о пользователе,

-данные о посте,

-лента постов в блоге,

-превью поста,

-текст поста.

3.3 Блок «Работа с комментариями»

Входящие дуги:

-данные о пользователе,

-данные о посте,

-текст комментария.

Дуги механизма:

-пользователи,

-адмнинистратор.

Дуги управления:

-правила блога,

-федеральный закон №152-ФЗ,

-федеральный закон №149-ФЗ.

Выходящие дуги:

-данные о пользователе,

-данные о посте,

-данные о комментариях,

-комментарии к посту.

3.4 Блок «Работа администратора»

Входящие дуги:

-данные о пользователе,

-данные о посте,

-данные о комментариях.

Дуги механизма:

-пользователи,

-адмнинистратор.

Дуги управления:

-правила блога,

-федеральный закон №152-ФЗ,

-федеральный закон №149-ФЗ.

Выходящие дуги:

-отмодерированный комментарий,

-отмодерированный пост,

-список пользователей.

4 Декомпозиция IDEF0 диаграммы 2 уровня в нотации DFD

Нотация DFD предназначена не только для описания процессов, а для описания потоков данных в ходе этих процессво. В нотации DFD была создана декомпозиция двух функциональных блоков из IDEF0 диаграммы первого уровня, а именно: работа с пользователем и работа с постами.

4.1 Декомпозиция блока «Работа с пользователем»

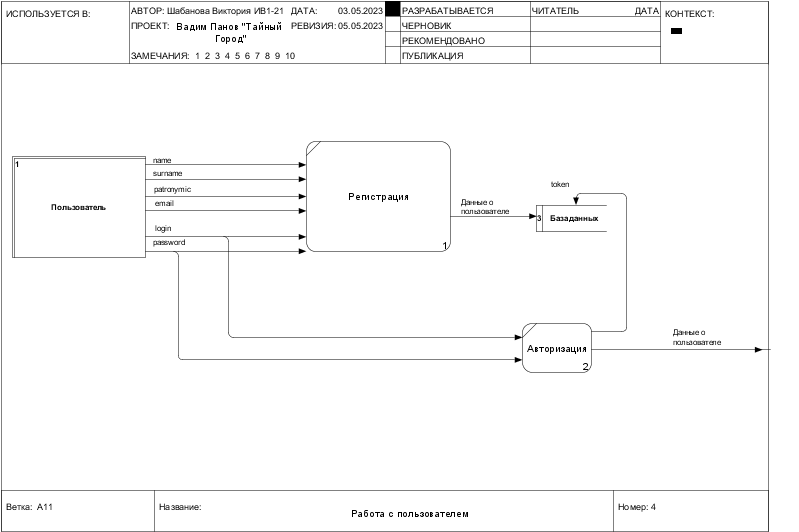
Декомпозиция этого блока представлена на рисунке 4. Блок «Работа с пользователем» содержит внешнюю сущность «Пользователь», которая обозначает пользователя разрабатываемого сайта. «Пользователь» передаёт в процесс «Регистрация» имя, фамилию, отчество, email, логин, пароль, а в процесс «Авторизация» только логин и пароль. Из процессов эта информация передаётся в базу данных. Также из процесса «Автризация» данные о пользователе переходят в следующий функциональный блок. 

Рисунок 4 – декомпозиция блока «Работа с пользователем» в нотации DFD

4.2 Декомпозиция блока «Работа с постами»

Декомпозиция этого блока представлена на рисунке 5. Из блока «Работа с пользователем» в данный блок передаются данные о пользователе, их принимают процессы «Редактирование поста» и «Создание поста». Также эти два процесса от внешней сущности «Пользователь» принимают текст поста, текст превью, текст названия. Процесс «Создание поста» передает данные о посте в базу данных, откуда они передаются в процессы «Редактирование поста» и «Удаление поста». В «Редактирование поста» «Пользователь» передает редактирование. И данные о посте из этого процесса передаются в базу данных. В «Удаление поста» «Пользователь» передаёт удаление. А из этого процесса в базу данных передаётся запрос на удаление.

В следующий функциональный блок из процессов «Редактирование поста» и «Создание поста» передается текст поста и данные о пользователе и данные о посте, из процесса «Удаление поста» передаются данные о посте.

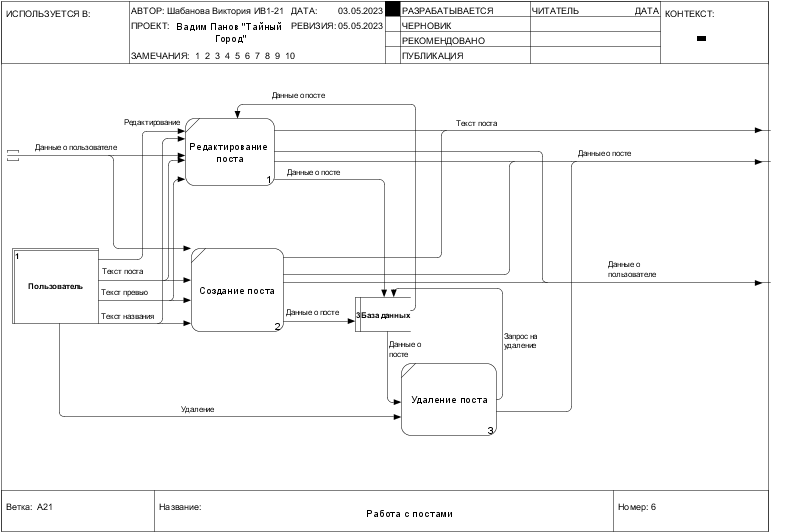


Рисунок 5 – декомпозиция блока «Работа с постами» в нотации DFD

5 Создание ER диаграммы базы данных

Для проектирования реляционной базы данных информационной системы нужно построить ER диаграмму, которая показывает таблицы и связи между ними. Созданная ER диаграмму показана на рисунке 6. В ней содержатся 4 таблицы, описание которых представлено в таблица 1-таблица 4.

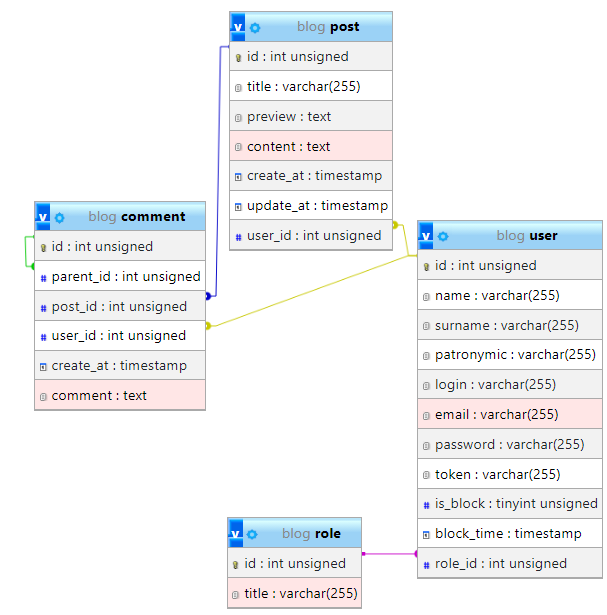


Рисунок 6 - ER диаграмма

Таблица 1 - Role

|  |  |  |  |
| --- | --- | --- | --- |
| Ключ | Поле | Тип | Ограничение |
| первичный | id | integer |  |
|  | title | varchar | 255 |

Таблица 2 - User

|  |  |  |  |
| --- | --- | --- | --- |
| Ключ | Поле | Тип | Ограничение |
| первичный | id | integer |  |
|  | name | varchar | 255 |
|  | surname | varchar | 255 |
|  | patronymic | varchar | 255 |
|  | login | varchar | 255 |
|  | email | varchar | 255 |
|  | password | varchar | 255 |
|  | token | varchar | 255 |
|  | is\_block | tinyint |  |
|  | block\_time | timestamp |  |
| внешний | role\_id | integer |  |

Связь «один-ко-многим» с таблицей Role через поле role\_id.

Таблица 3 – Post

|  |  |  |  |
| --- | --- | --- | --- |
| Ключ | Поле | Тип | Ограничение |
| первичный | id | integer |  |
|  | title | varchar | 255 |
|  | preview | text |  |
|  | content | text |  |
|  | create\_at | timestamp |  |
|  | update\_at | timestamp |  |
| внешний | user\_id | integer |  |

Связь «один-ко-многим» с таблицей User через поле user\_id.

Таблица 4 – Comment

|  |  |  |  |
| --- | --- | --- | --- |
| Ключ | Поле | Тип | Ограничение |
| первичный | id | integer |  |
| внешний | parent\_id | integer |  |
| внешний | post\_id | integer |  |
| внешний | user\_id | integer |  |
|  | create\_at | timestamp |  |
|  | comment | text |  |

Связь «один-ко-многим» с таблицей User через поле user\_id. Связь «один-ко-многим» с таблицей Post через поле post\_id. Рекурсивная связь через поле parent\_id.

6 Создание диаграммы компонентов системы

В ходе процесса проектирования информационной системы была разработана диаграмма компонентов, которая реализует основной функционал разрабатываемой системы. Она отражает классы, которые необходимо использовать в процессе разработки, связи и зависимости между ними, методы и атрибуты, содержащиеся в них. Диаграмма представлена на рисунке 7.

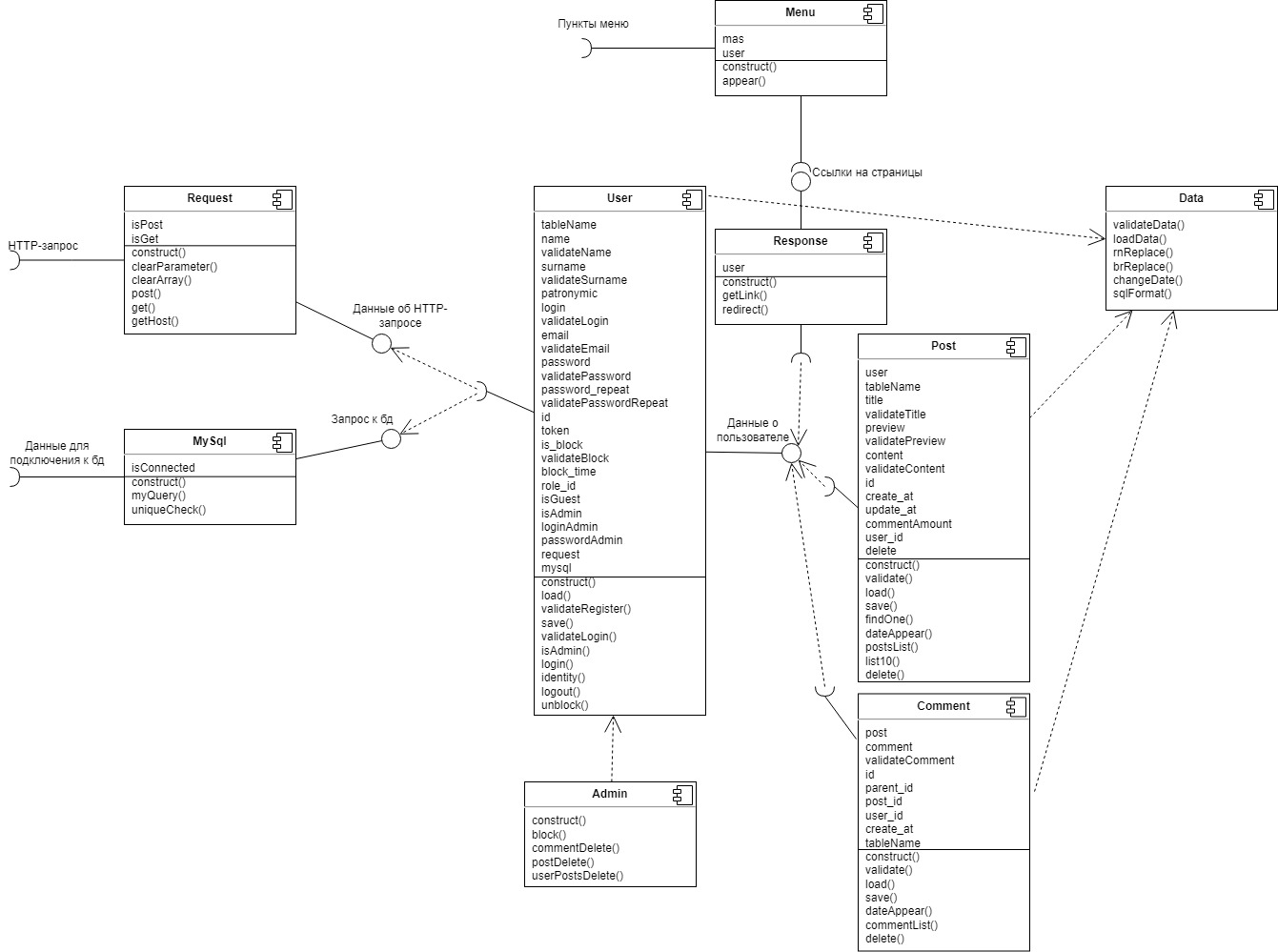


Рисунок 7 – диаграмма компонентов

Класс «Request» через требуемый интерфейс принимает HTTP-запрос и передает данные об HTTP-запросе через предоставляемый интерфейс.

Класс «MySql» через требуемый интерфейс принимает массив с данными для подключения к базе данных и передает запрос к базе данных через предоставляемый интерфейс.

Класс «User» принимает через требуемый интерфейс данные об HTTP-запросе и запрос к базе данных от классов «Request» и «MySql» соответственно. И передает данные о пользователе через предоставляемый интерфейс. Класс зависит от класса «Data».

Класс «Admin» зависит от класса «User».

Класс «Response» принимает через требуемый интерфейс данные о пользователе от класса «User» и передает ссылки на страницы через предоставляемый интерфейс.

Класс «Menu» принимает через требуемый интерфейс массив с пунктами меню и ссылки на страницы от класса «User».

Класс «Post» зависит от класса «Data». Класс принимает через требуемый интерфейс данные о пользователе от класса «User».

Класс «Comment» зависит от класса «Data». Класс принимает через требуемый интерфейс данные о пользователе от класса «User».

7 Реализация информационной системы

Написание программного кода для информационной системы проводится поэтапно.

7.1 Подготовка к реализации функций информационной системы

В директории с проектом «info-system» создается отдельная директория «lib» для подключаемых файлов и классов. В ней создается файл конфигурации приложения, autoloader для автоматического подключения классов, файл инициализации приложения, в котором подключается файл конфигурации и autoloader. В файле инициализации создаются объекты всех классов.

В файле конфигурации создаются два массива. Массив $sql\_conf, в котором указываются параметры подключения к базе данных. Он представлен на рисунке 8.

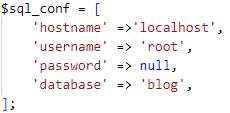


Рисунок 8 – массив $sql\_conf

Второй массив - $menu\_conf, его элементы – это массивы, в которых ключ – название пункта меню, значение – адрес страницы сайта. Он представлен на рисунке 9.

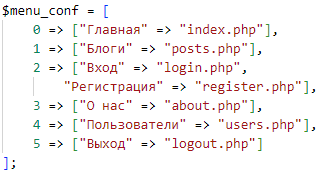


Рисунок 9 – массив $menu\_conf

В директории «info-system» хранятся файлы с HTML-разметкой для каждой страницы. Файлы инициализации для каждой из этих страниц, которые подключаются к ним, находятся в «lib». Внутри этих файлов подключается файл инициализации приложения.

7.2 Создание класса Request

Класс Request нужен для обработки HTTP-запросов. Он содержит свойства $isGet и $isPost, которые указывают на то, какой метод у HTTP-запроса.

7.2.1 Конструктор

Конструктор класса представлен на рисунке 10. В нём инициализируются атрибуты isGet и $isPost, через обращение к массиву $\_SERVER.

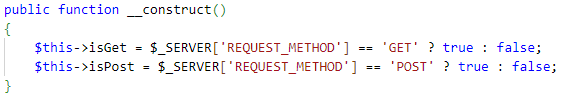


Рисунок 10 – конструктор класса

7.2.2 Метод clearParameter

Этот метод предназначен для обработки входных параметров HTTP-запроса, из него с помощью встроенных функций trim и strip\_tags удаляются лишние пробелы в начале и конце, а также HTML-теги, для осуществления базовой защиты информационной системы. Сам метод представлен на рисунке 11.

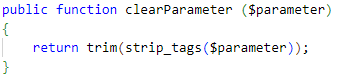


Рисунок 11 – метод clearParameter

7.2.3 Метод clearArray

Этот метод принимает массив, элементы которого обрабатываются с помощью метода clearParameter, описанного в пункте 7.2.2, и возвращает обработанный массив. Если элементы входного параметра тоже массивы, то метод вызывается рекурсивно. Метод представлен на рисунке 12.

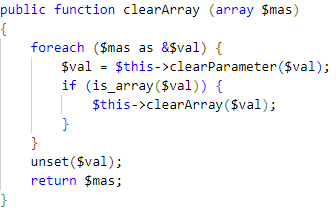


Рисунок 12 – метод clearArray

7.2.4 Метод post

Этот метод нужен для обработки данных, которые отправляются POST-запросом. У него есть необязательный параметр. Если он передан, то он передается в метод clearParameter, описанный в 7.2.2. Если он не передан, то вызывается метод clearArray, описанный в 7.2.3, и с помощью него обрабатывается массив $\_POST. Метод post представлен на рисунке 13.

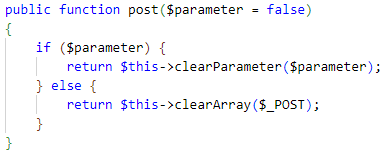


Рисунок 13 – метод post

7.2.5 Метод get

Этот метод нужен для обработки данных, которые отправляются GET-запросом. У него есть необязательный параметр. Если он не передан, то вызывается метод clearArray, описанный в 7.2.3, и с помощью него обрабатывается массив $\_GET. Если параметр передан, и он есть в массиве $\_GET, то он передается в метод clearParameter, описанный в 7.2.2, а если параметра в массиве нет, то метод возвращает null. Метод get представлен на рисунке 14.

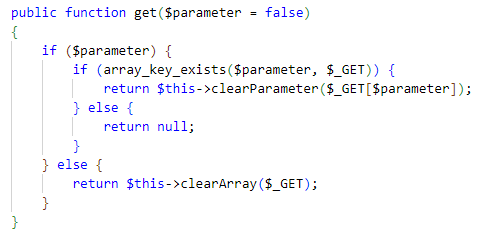


Рисунок 14 – метод get

7.2.6 Метод getHost

Этот метод обращается к массиву $\_SERVER и возвращает название хоста. Метод представлен на рисунке 15.

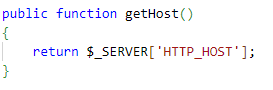


Рисунок 15 – метод getHost

7.3 Создание класса MySql

Этот класс нужен для работы с базой данных, он наследуется от встроенного класса mysqli. Класс содержит атрибут $isConnected, который является признаком наличия соединения с сервером базы данных.

7.3.1 Конструктор

В конструктор передается массив $sql\_conf из файла конфигурации приложения. Внутри вызывается родительский конструктор, который осуществляет подключение к базе данных. Для инициализации атрибута $isConnected вызывается родительский метод ping. Конструктор класса представлен на рисунке 16.

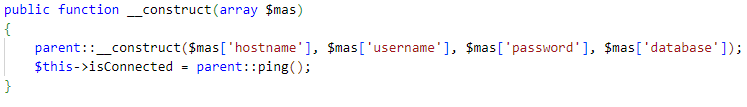


Рисунок 16 – конструктор класса

7.3.2 Метод myQuery

Этот метод принимает текст запроса к базе данных, осуществляет его с помощью вызова родительского метода query и возвращает либо логический результат запроса, либо ассоциативный массив с возвращенными полями. Если атрибут $isConnected указывает на разрыв соединения с сервером, то метод возвращает false. Сам метод представлен на рисунке 17.

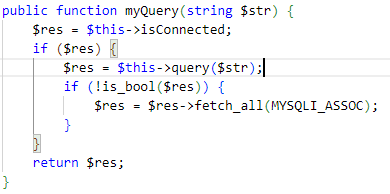


Рисунок 17 – метод myQuery

7.3.3 Метод uniqueCheck

Этот метод вызывает myQuery, описанный в пункте 7.3.2, и передает туда запрос на проверку уникальности определенного значения в указанном поле указанной таблицы. Возвращает true, если указанное значение не встречается в таблице. Метод представлен на рисунке 18.

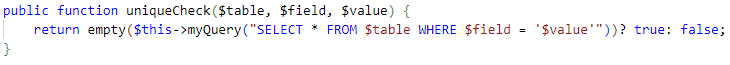


Рисунок 18 – метод uniqueCheck

7.4 Создание класса Data

Этот класс предназначен для универсальной обработки данных.

7.4.1 Метод validateData

Этот метод возвращает признак сработало или нет хотя-бы одно из правил валидации в любом классе, которые он наследует. В методе перебираются все атрибуты объекта текущего класса и проверяется, есть ли значение у атрибутов валидации. Если хотя бы у одного атрибута заполнено значение, значит валидация не прошла и метод возвращает false, в противном случае возвращает true. Сам метод представлен на рисунке 19.

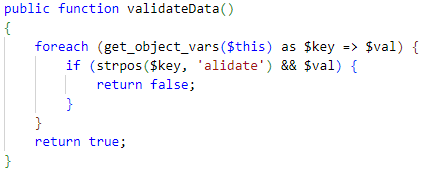


Рисунок 19 – метод validateData

7.4.2 Метод loadData

Данный метод принимает массив данных, если в текущем классе у атрибутов названия совпадают с ключами в массиве, то атрибутам присваиваются значения из массива и возвращается true, если передан пустой массив, возвращается false. Метод представлен на рисунке 20.

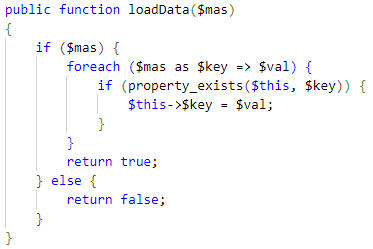


Рисунок 20 – метод loadData

7.4.3 Метод rnReplace

Этот метод принимает строку и все вхождения «\r\n» заменяет на тег <br/> и возвращает измененную строку. Он представлен на рисунке 21.

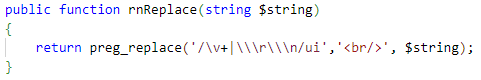


Рисунок 21 – метод rnReplace

7.4.4 Метод brReplace

Этот метод принимает строку и все вхождения тега <br/> заменяет на «\r\n» и возвращает измененную строку. Он представлен на рисунке 22.

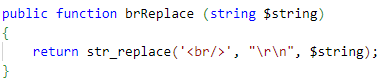


Рисунок 22 – метод brReplace

7.4.5 Метод changeDate

Этот метод принимает строку и с помощью встроенного класса DateTimeImmutable меняет формат введенной даты на формат, соответствующий русской локализации, возвращает дату в новом формате. Метод представлен на рисунке 23.

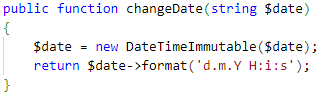


Рисунок 23 – метод changeDate

7.4.6 Метод sqlFormat

Этот метод принимает строку и с помощью встроенного класса DateTimeImmutable меняет формат введенной даты на формат даты, с которым работает язык запросов SQL, возвращает дату в новом формате. Метод представлен на рисунке 24.

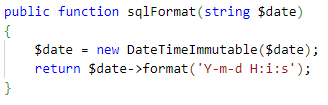


Рисунок 24 – метод sqlFormat

7.5 Создание класса User

Этот класс реализует работу с пользователями системы, он наследуется от класса Data, описанного в пункте 7.4. В нём содержатся атрибуты:

- название таблицы из базы данных - $tablename, по умолчанию равен «user»;

- поля формы регистрации, авторизации: $name, $surname, $patronymic – по умолчанию null, $login, $email, $password, $password\_repeat;

- валидация полей регистрации, авторизации, блокировки пользователя: $validateName, $validateSurname, $validateLogin, $validateEmail, $validatePassword, $validatePasswordRepeat, $validateBlock;

- столбцы таблицы: $id, $token, $is\_block, $block\_time, $role\_id;

- логин администратора - $loginAdmin, по умолчанию равен «admin»;

- пароль администратора - $passwordAdmin, по умолчанию равен «admin»;

- признак авторизации пользователя - $isGuest, по умолчанию равен true;

- признак роли администратора - $isAdmin;

- экземпляр класса Request - $request;

- экземпляр класса MySql - $mysql.

7.5.1 Конструктор

Конструктор принимает объекты классов Request и MySql, которые были описаны в пунктах 7.2 и 7.3 соответственно. Объекты этих классов присваиваются атрибутам класса User. Также в конструкторе идет проверка с помощью метода get класса Request, описанного в пункте 7.2.5, передает ли GET-запрос параметр токен, если он передан, то вызывается метод identity, который будет рассмотрен дальше. Сам конструктор представлен на рисунке 25.

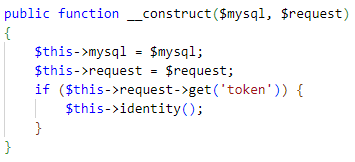


Рисунок 25 – конструктор

7.5.2 Метод isAdmin

Этот метод проверяет, совпадает ли логин пользователя в атрибуте $login и пароль в атрибуте $password с логином и паролем администратора, хранящимися в атрибутах $loginAdmin и $passwordAdmin. Если они совпадают, то метод возвращает true, в противном случае false. Сам метод представлен на рисунке 26.

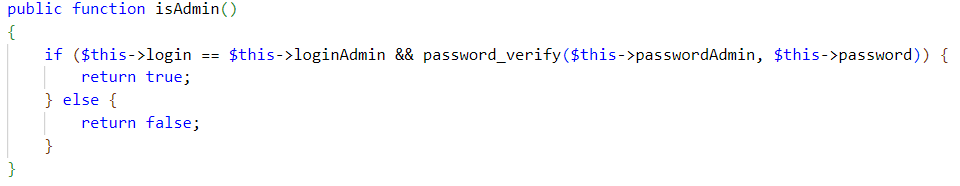


Рисунок 26 – метод isAdmin

7.5.3 Метод load

Этот метод принимает массив данных, передает его в родительский метод loadData, описанный в пункте 7.4.2, и возвращает его результат. Также внутри метода load инициализируется атрибут $isAdmin через метод isAdmin. Сам метод load представлен на рисунке 27.

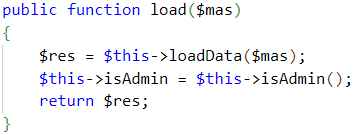


Рисунок 27 – метод load

7.5.4 Метод validateRegister

Метод создан для валидации данных при регистрации. В нём проверяется соответствие требованиям у атрибутов, которые содержат значение полей формы регистрации. В случае несоответствия требованиям в атрибут валидации записывается сообщение об ошибке. Далее вызывается родительский метод validateData, описанный 7.4.1, и возвращается его значение. Сам метод validateRegister представлен на рисунке 28.



Рисунок 28 – метод validateRegister

7.5.5 Метод save

Этот метод сохраняет в базу данных значения атрибутов с данными для регистрации, при этом пароль передается в захешированном виде. Сам метод представлен на рисунке 29.

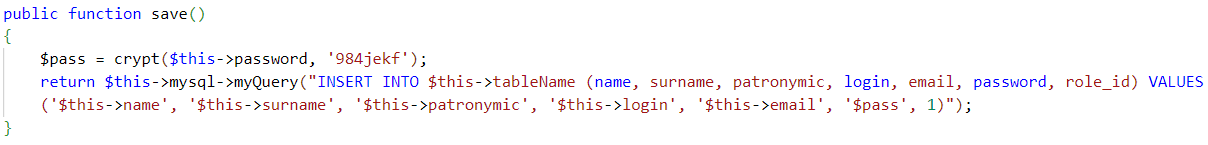


Рисунок 29 – метод save

7.5.6 Метод validateLogin

Метод создан для валидации данных, вводимых при входе пользователя в систему. В нём проверяется соответствие требованиям у атрибутов, которые содержат значение полей формы авторизации. В случае несоответствия требованиям в атрибут валидации записывается сообщение об ошибке. Далее вызывается родительский метод validateData, описанный 7.4.1, и возвращается его значение. Сам метод validateLogin представлен на рисунке 30.

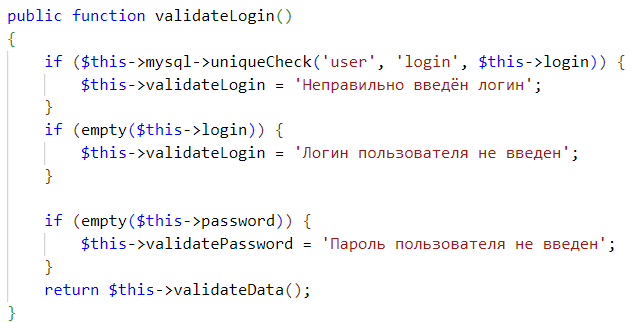


Рисунок 30 – метод validateLogin

7.5.7 Метод login

Этот метод осуществляет вход пользователя в систему. Сначала проверяется соответствует ли введенный пароль введенному логину, которые хранятся в атрибутах $password и $login. Если пароль не правильный, то метод возвращает false и атрибуту $validatePassword присваивается информация об ошибке. Если пароль правильный, то с помощью метода load, описанного в пункте 7.5.3, в атрибуты объекта передаются значения столбцов таблицы с данными о авторизованном пользователе. Далее через атрибут $is\_block проверяется заблокирован ли пользователь. Если пользователь заблокирован, то в атрибут $validateBlock записываются данные об ошибке, и метод возвращает false. Если пользователь не заблокирован, то атрибуту $isGuest присваивается значение false, формируется токен для аккаунта пользователя, записывается в базу данных, метод возвращает true. Сам метод представлен на рисунке 31.



Рисунок 31 – метод login

7.5.8 Метод identity

Этот метод принимает необязательный параметр $id, который указывает на id пользователя в базе данных. Если он передан, то по нему из базы данных с помощью метода load, описанного в пункте 7.5.3, загружаются значения столбцов в атрибуты объекта. Если $id не передан, то через метод get, описанный в пункте 7.2.5, достается токен пользователя и данные загружаются с помощью него. Если такого токена нет в базе данных, то метод возвращает false. Если токен есть, атрибуту $isGuest присваивается значение false, и метод возвращает true. Сам метод представлен на рисунке 32.

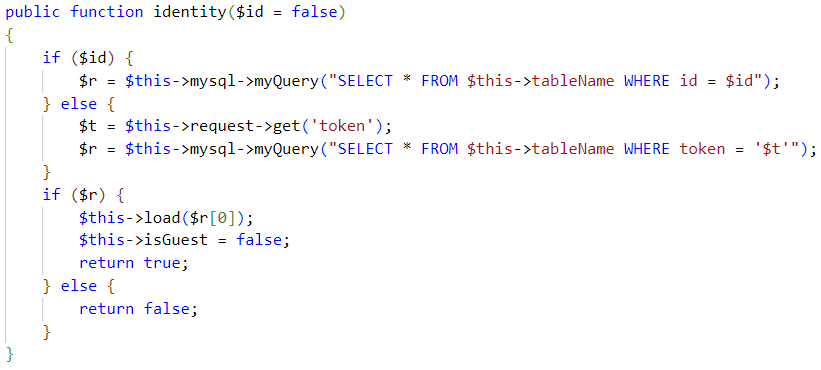


Рисунок 32 – метод identity

7.5.8 Метод logout

Этот метод осуществляет выход пользователя из системы, если он был авторизован. Для этого из базы данных и из атрибута $token удаляется значение токена, а атрибуту $isGuest присваивается значение true, и метод возвращает true. Если пользователь не был авторизован, то метод возвращает false. Сам метод представлен на рисунке 33.



Рисунок 33 – метод logout

7.5.9 Метод unblock

Этот метод предназначен для снятия блокировки с временно заблокированного пользователя. Для этого из базы данных с помощью метода myQuery, описанного в пункте 7.3.2, извлекаются все заблокированные пользователи. После чего проверяется закончилось ли время блокировки пользователя, для чего оно сравнивается с текущей датой. Если время блокировки закончилось, то в базе данных меняется значение столбца is\_block на 0. Сам метод представлен на рисунке 34.

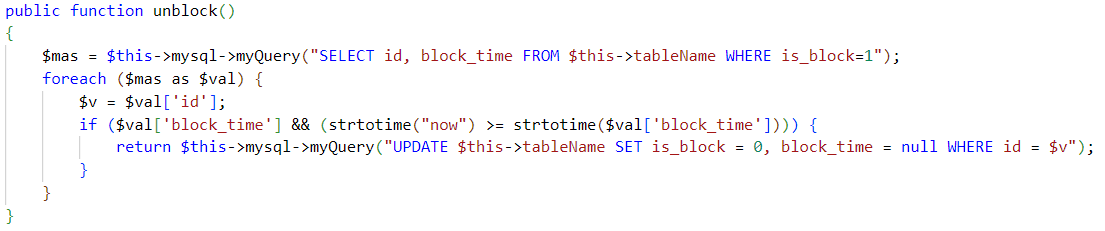


Рисунок 34 – метод unblock

7.6 Создание класса Response

Класс Response необходим для формирования ссылок и перенаправления клиента на необходимый адрес. У него есть атрибут $user, куда помещается объект класса User, описанного в пункте 7.5

7.6.1 Конструктор

Конструктор класса принимает объект класса User и присваивает его атрибуту $user. Также если в GET-запросе передаётся токен пользователя, но токена нет в атрибуте объекта, то пользователя перенаправляет на главную страницу. Сам конструктор представлен на рисунке 35.

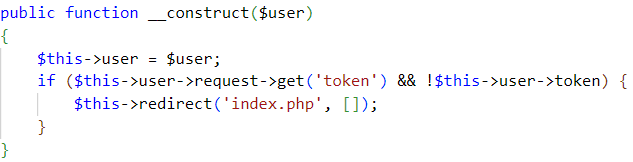


Рисунок 35 – конструктор

7.6.2 Метод getLink

Этот метод нужен для формирования конечного варианта ссылки для

перехода на страницу. Метод принимает url и массив с параметрами GET-запроса. В методе производится проверка если пользователь авторизован и в массиве параметров отсутствует токен, то он добавляется в массив из атрибута $token объекта класса User. Далее производится проверка на наличие знака «?» в url. В случае если массив параметров не пустой и знак «?» отсутствует, он добавляется. После чего элементы массива записываются в конец url в формате «параметр=значение», между собой соединяются знаком «&» Метод возвращает готовый url. Сам метод представлен на рисунке 36.

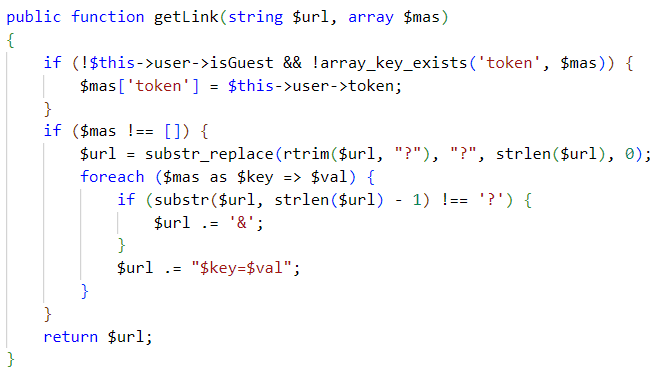


Рисунок 36 – метод getLink

7.6.3 Метод redirect

Это метод для перенаправления клиента на определенную страницу. Он принимает url и массив с параметрами GET-запроса. В методе вызывается метод getLink, описанный в пункте 7.6.2, в него передается url и массив с параметрами GET-запроса. После формируется окончательная ссылка для перенаправления, для этого указывается протокол, имя хоста из метода getHost, описанного в пункте 7.2.6, директория проекта и ссылка, сформированная через метод getLink. Сам метод представлен на рисунке 37.

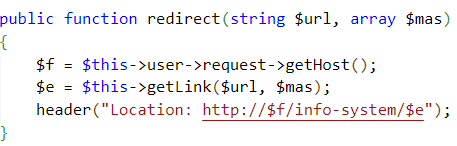


Рисунок 37 – метод redirect

7.7 Реализация регистрации пользователя

Для настройки формы регистрации указывается файл для обработки формы и метод запроса для ее передачи. Посте этого для каждого поля формы создаётся условие отображения класса ошибки, если для этого поля введено неверное значение. По умолчанию для полей формы ставится значение атрибутов класса User, описанного в пункте 7.5, куда записываются данные отправленной формы. В случае неверно введенных данных под полем отображаются атрибуты валидации полей из класса User. Пример создания одного из полей формы регистрации представлен на рисунке 38.

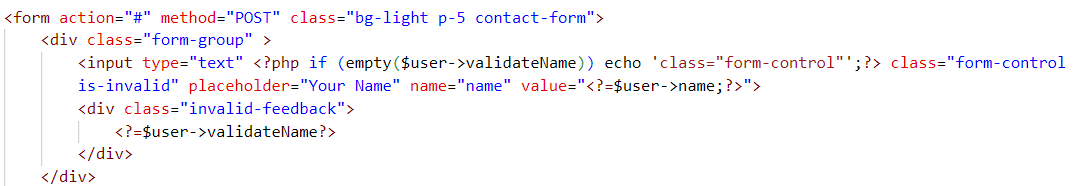


Рисунок 38 – поле формы регистрации

После этого в файле инициализации страницы регистрации производится проверка на метод запроса POST с помощью класса Request, описанного в пункте 7.2. Через метод load, описанный в пункте 7.5.3, в объект класса User загружаются значения полей формы регистрации, они проверяются через метод validateRegister, описанный в пункте 7.5.4. Если валидация прошла успешно, то данные о новом пользователе сохраняются в базу данных с помощью метода save, описанного в пункте 7.5.5, пользователь входит в систему с помощь метода login, описанного в пункте 7.5.7, и переходит на главную страницу c помощью метода redirect, описанного в пункте 7.6.3. Сам файл инициализации страницы регистрации представлен на рисунке 39.

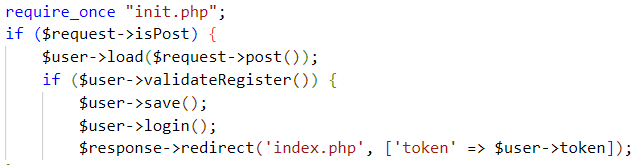


Рисунок 39 – файл инициализации формы регистрации

7.8 Реализация авторизации пользователя

В форме авторизации указывается файл обработки формы и метод запроса. Для каждого поля создаётся условие отображения класса ошибки, если для этого поля введено неверное значение. По умолчанию для полей формы ставится значение атрибутов класса User, куда записываются данные отправленной формы. Под полем отображаются атрибуты валидации полей из класса User. Поля формы авторизации представлены на рисунке 40.



Рисунок 40 – поля формы авторизации

После этого в файле инициализации страницы авторизации производится проверка на метод запроса POST с помощью класса Request, описанного в пункте 7.2. Через метод load, описанный в пункте 7.5.3, в объект класса User загружаются значения полей формы авторизации, они проверяются через метод validateLogin, описанный в пункте 7.5.6. Если валидация прошла успешно, то вызывается метод unblock, описанный в пункте 7.5.9. После этого пользователь входит в систему с помощь метода login, описанного в пункте 7.5.7. Если пользователь успешно вошел, то он переходит на главную страницу c помощью метода redirect, описанного в пункте 7.6.3. Сам файл инициализации страницы авторизации представлен на рисунке 41.



Рисунок 41 – файл инициализации страницы авторизации

7.9 Создание меню информационной системы

Для создания меню создается специальный класс Menu. Его метод appear вызывается на каждой HTML-странице сайта.

7.9.1 Конструктор

В классе Menu есть атрибут $mas, в котором содержится массив с пунктами меню, $response, в котором содержится объект класса Response, описанного в пункте 7.6. В конструкторе происходит инициализация этих атрибутов. Сам конструктор представлен на рисунке 42.

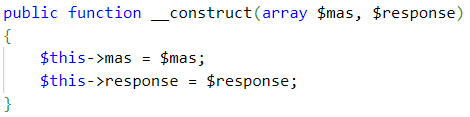


Рисунок 42 – конструктор

7.9.2 Метод appear

Этот метод отображает HTML-код меню. В него передаётся название файла страницы, на которой сейчас находится пользователь. В нем в зависимости от роли пользователя в формируемый код добавляются разные пункты меню и в зависимости от страницы, на которой находится пользователь пунктам меню присваивается класс подсветки. Метод возвращает строчку с HTML-кодом меню. Сам метод представлен на рисунке 43.



Рисунок 43 –метод appear

7.10 Создание класса Post

Этот класс реализует функции работы с постами на сайте, он наследуется от класса Data, описанного в пункте 7.4. В нём содержатся атрибуты:

- экземпляр класса User – $user;

- название таблицы из базы данных – $tablename, по умолчанию равен «post»;

- поля формы создания/редактирования поста: $title, $preview, $content;

- валидация полей формы создания/редактирования поста: $validateTitle, $validatePreview, $validateContent, $delete;

- столбцы таблицы: $id, $create\_at, $update\_at, $user\_id;

- количество комментариев - $commentAmount.

В конструкторе инициализируется атрибут $user. Сам конструктор представлен на рисунке 44.

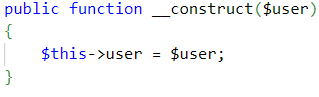


Рисунок 44 – конструктор

7.10.1 Метод validate

Метод создан для валидации данных при создании поста. В нём проверяется соответствие требованиям у атрибутов, которые содержат значение полей формы создания поста. В случае несоответствия требованиям в атрибут валидации записывается сообщение об ошибке. Далее вызывается родительский метод validateData, описанный 7.4.1, и возвращается его значение. Сам метод validate представлен на рисунке 45.

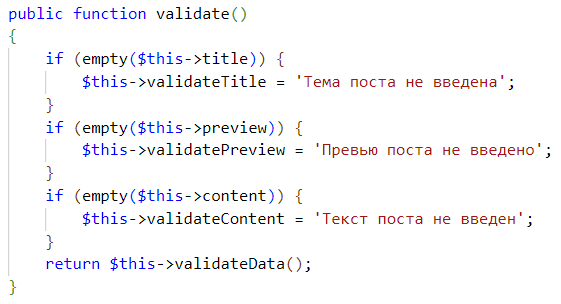


Рисунок 45 – метод validate

7.10.2 Метод load

Этот метод принимает массив данных, передает его в родительский метод loadData, описанный в пункте 7.4.2, и возвращает его результат. Сам метод load представлен на рисунке 46.

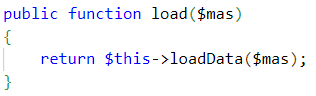


Рисунок 46 – метод load

7.10.3 Метод save

Этот метод сохраняется в базу данных информацию о созданном посте. Причем текст поста преобразуется через родительский метод rnReplace, описанный в пункте 7.4.3. С помощью метода get, описанного в пункте 7.2.5, проверяется не передан ли через GET-запрос параметр «post». Если он передан, то в базе данных не создается новая запись, а обновляются данные в поле запроса, id которого передан в GET-запросе. Сам метод представлен на рисунке 47.

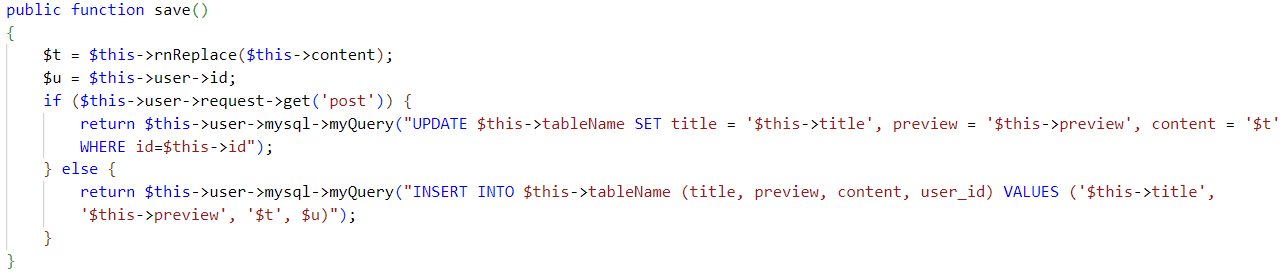


Рисунок 47 – метод save

7.10.4 Метод findOne

Этот метод принимает параметр $id, который указывает на id поста в базе данных. Если такой пост есть, то из базы данных с помощью метода load, описанного в пункте 7.10.3, загружаются значения столбцов в атрибуты объекта. Также в атрибут $commentAmont записывается количество комментариев к посту, которое вычисляется с помощью запроса к базе данных. После чего метод возвращает true. Если такой пост не был найдет, то метод возвращает false. Сам метод представлен на рисунке 48.

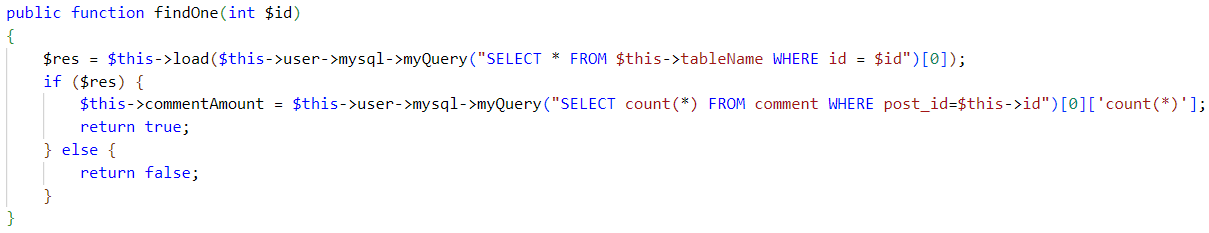


Рисунок 48 – метод findOne

7.10.5 Метод dateAppear

Этот метод нужен для изменения формата даты. Внутри него вызывается родительский метод changeDate, описанный в пункте 7.4.5. Сам метод представлен на рисунке 49.

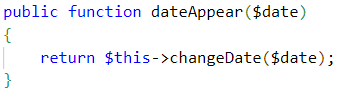


Рисунок 49 – метод dateAppear

7.10.6 Метод postsList

Этот метод формирует список постов. Он принимает необязательный параметр $limit, определяющий количество возвращаемых записей. К базе данных делается запрос, который возвращает информацию о постах и количество комментариев к посту, посты отсортированы от новых к старым. Создается массив объектов текущего класса, в каждый объект загружается информация о посте, получаемая из базы данных. Метод возвращает массив объектов. Сам метод postsList представлен на рисунке 50.

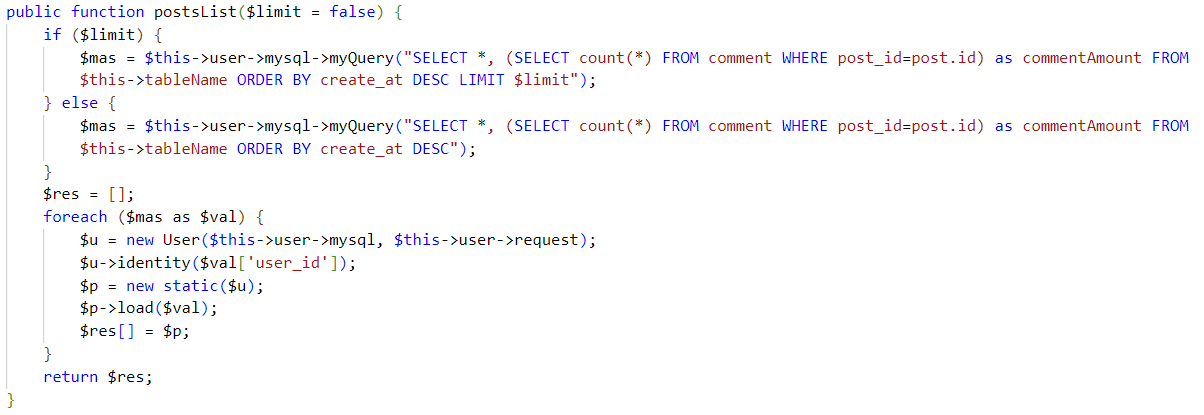


Рисунок 50 – метод postsList

7.10.7 Метод list10

Этот метод возвращает информацию о 10 последних добавленных постах. Для этого внутри него вызывается метод postsList, описанный в пункте 7.10.7, куда передается число 10. Сам метод list10 представлен на рисунке 51.

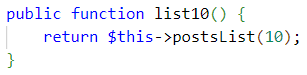


Рисунок 51 – метод list10

7.10.8 Метод delete

Этот метод принимает id поста и удаляет его, посылая запрос к базе данных с помощью метода myQuery, описанного в пункте 7.3.2. Сам метод представлен на рисунке 52.

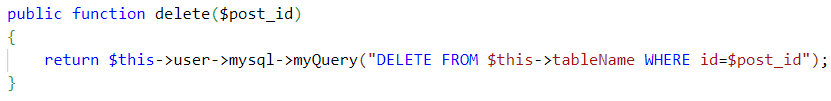


Рисунок 52 – метод delete

7.11 Реализация функции создания и редактирования поста

Для настройки формы создания/редактирования поста указывается файл для обработки формы и метод запроса для ее передачи. Далее для каждого поля создаётся условие отображения класса ошибки, если для этого поля введено неверное значение. По умолчанию для полей формы ставится значение атрибутов класса Post, куда записываются данные отправленной формы. При этом содержимое атрибута $content отображается с помощью метода brReplace, описанного в пункте 7.4.4. В случае неверно введенных данных под полем отображаются атрибуты валидации полей из класса Post.

Кроме кнопки отправки поста на странице есть кнопка удаления поста. Она перенаправляет пользователя на страницу текущего поста с указанием в GET-запросе id поста, который нужно удалить. Ссылка для перенаправления формируется с помощь метода getLink, описанного в пункте 7.6.2.

Сама форма создания/редактирования поста представлены на рисунке 53.



Рисунок 53 – форма создания/редактирования поста

В файле инициализации страницы создания/редактирования поста производится проверка, что пользователь авторизован и при этом не является администратором, проверка на метод запроса POST с помощью класса Request, описанного в пункте 7.2. Через метод load, описанный в пункте 7.10.3, в объект класса Post загружаются значения полей формы авторизации, они проверяются через метод validate, описанный в пункте 7.10.2 Если валидация прошла успешно, то данные сохраняются в базу данных через метод save, описанный в пункте 7.10.4. Если у объекта класса Post не задан $id, то через метод load в атрибуты объекта загружаются значения столбцов у последнего добавленного поста.

При редактировании поста в GET-запросе через параметр «post» передается id редактируемого поста. Поэтому в файле инициализации создания/редактирования поста через метод get, описанный в пункте 7.2.5, делается проверка передан ли нужный параметр. Если он передан, то по нему в атрибуты объекта класса Post загружаются данные о посте с помощью метода findOne, описанного в пункте 7.10.5. Сам файл инициализации страницы создания/редактирования поста представлен на рисунке 54.



Рисунок 54 – файл инициализации страницы создания/редактирования поста

7.12 Отображения постов на главной странице

На главной станице должны отображаться последние 10 добавленных постов. Для получения информации по ним вызывается метод list10, описанный в пункте 7.10.8, он возвращает массив. После чего для каждого отдельного элемента полученного массива формируется ссылка на соответствующий пост с помощью метода getLink, описанного в пункте 7.6.2, туда передается ссылка на страницу просмотра поста, токен пользователя, если он есть, и id поста.

Для отображения одного поста на странице формируется HTML-код, куда добавляются атрибуты класса POST, в которых содержится логин пользователя, название, количество комментариев, превью поста. Дата создания поста отображается через метод changeDate, описанный в пункте 7.4.5. Сформированная ссылка добавляется в атрибут href у названия поста и надписи «Подробнее».

Сам код отображения поста на главной странице представлен на рисунке 55.



Рисунок 55 – отображение списка постов на главной странице

7.13 Отображение списка постов на странице со всеми постами

На странице просмотра постов кроме их списка должна отображаться кнопка создания поста. Ссылка на форму создания поста формируется с помощью метода getLink, описанного в пункте 7.6.2. Туда передается название файла и токен пользователя. Сама кнопка отображается только если пользователь авторизован и при это не является администратором. Сам код отображения кнопки представлен на рисунке 56.

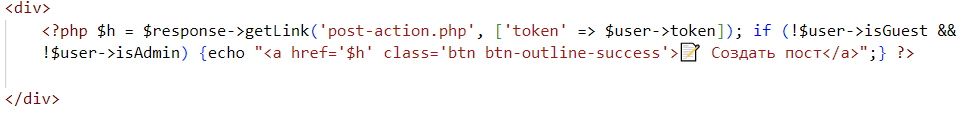


Рисунок 56 – отображения кнопки создания поста

Для получения информации о постах вызывается метод postsList, описанный в пункте 7.10.7, он возвращает массив. После чего для каждого отдельного элемента полученного массива формируется ссылка на соответствующий пост с помощью метода getLink, описанного в пункте 7.6.2, туда передается ссылка на страницу просмотра поста, токен пользователя, если он есть, и id поста. Также формируется ссылка для удаления поста, для этого кроме перечисленных параметров в getLink добавляется параметр delete со значением id поста. Также формируется ссылка на редактирование поста с помощью метода getLink, туда передается ссылка на страницу редактирования поста, токен пользователя и id поста.

Для отображения одного поста на странице формируется HTML-код, куда добавляются атрибуты класса POST, в которых содержится логин пользователя, название, количество комментариев, превью поста. Дата создания поста отображается через метод changeDate, описанный в пункте 7.4.5. Сформированная ссылка на просмотр поста добавляется в атрибут href у названия поста и надписи «Подробнее».

Рядом с каждым постом есть кнопки удаления и редактирования поста. Кнопка удаления появляется, если пользователь является администратором, в ее атрибут href передается ранее сформированная ссылка на удаление поста. Кнопка редактирования поста появляется если пользователь авторизован и $id авторизованного пользователя совпадает с $user\_id поста. В ее атрибут href передается ранее сформированная ссылка на редактирование.

Сам код отображения списка постов на странице просмотра всех постов представлен на рисунке 57.



Рисунок 57 – отображение списка всех постов

7.14 Создание класса Comment

Этот класс реализует функции работы с комментариями на сайте, он наследуется от класса Data, описанного в пункте 7.4. В нём содержатся атрибуты:

- экземпляр класса Post – $post;

- название таблицы из базы данных – $tablename, по умолчанию равен «post»;

- поле формы создания комментария: $comment;

- валидация поля формы создания комментария: $validateComment;

- столбцы таблицы: $id, $parent\_id, $create\_at, $user\_id, $post\_id;

В конструкторе инициализируется атрибут $post. Сам конструктор представлен на рисунке 58.

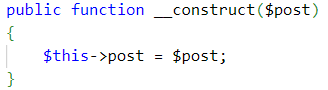


Рисунок 58 – конструктор

7.14.1 Метод validate

В методе проверяется было ли заполнено поле создания комментария. Если нет, в атрибут валидации записывается сообщение об этом. Далее вызывается метод validateData, описанный 7.4.1, и возвращается его значение. Сам метод validate представлен на рисунке 59.

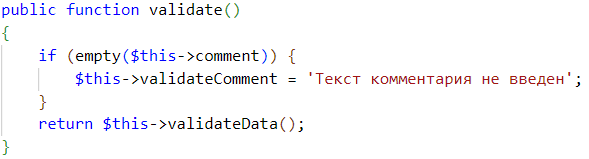


Рисунок 59 – метод validate

7.14.2 Метод load

Он принимает массив данных, передает его в метод loadData, описанный в пункте 7.4.2. Сам метод load представлен на рисунке 60.

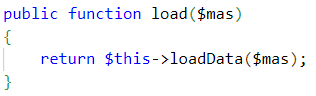


Рисунок 60 – метод load

7.14.3 Метод save

Этот метод сохраняется в базу данных информацию об оставленном комментарии. Причем текст преобразуется через родительский метод rnReplace, описанный в пункте 7.4.3. Сам метод представлен на рисунке 61.

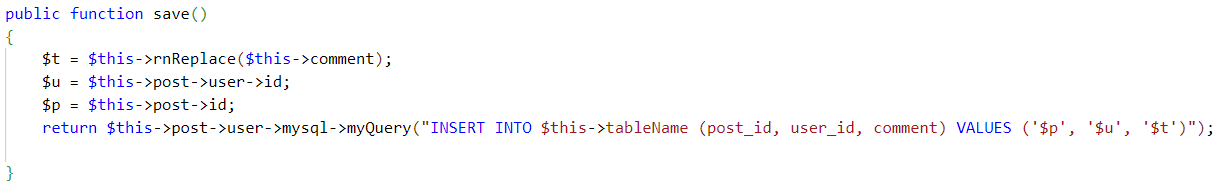


Рисунок 61 – метод save

7.14.4 Метод dateAppear

Этот метод нужен для изменения формата даты. Внутри него вызывается родительский метод changeDate, описанный в пункте 7.4.5. Сам метод представлен на рисунке 62.

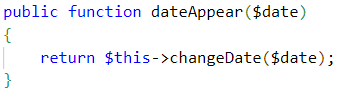


Рисунок 62 – метод dateAppear

7.14.5 Метод commentList

Этот метод формирует список комментариев. Он принимает параметр $post\_id, указывающий на пост, к которому оставлены комментарии. К базе данных делается запрос, который возвращает информацию о комментариях к нужному посту. Создается массив объектов текущего класса, в каждый объект загружается информация о комментарии, получаемая из базы данных. Метод возвращает массив объектов. Сам метод commentList представлен на рисунке 63.

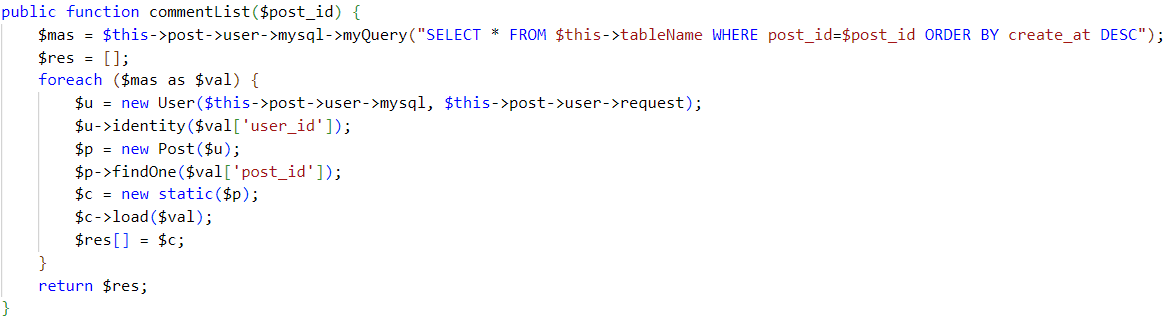


Рисунок 63 – метод commentList

7.14.6 Метод delete

Этот метод принимает id комментария и удаляет его, посылая запрос к базе данных с помощью метода myQuery, описанного в пункте 7.3.2. Сам метод представлен на рисунке 64.

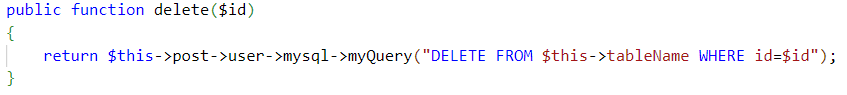


Рисунок 64 – метод delete

7.15 Создание класса Admin

Этот класс предназначен для реализации функций работы администратора. Он наследуется от класса User, описанного в пункте 7.5. В его конструктор передаются объекты класса MySql, описанного в пункте 7.3, и Request, описанного в пункте 7.2. Они передаются в вызванный родительский конструктор, описанный в пункте 7.5.1. Сам конструктор представлен на рисунке 65.

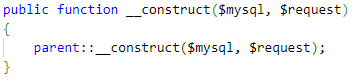


Рисунок 65 – конструктор

7.15.1 Метод block

Этот метод предназначен для реализации функции блокировки пользователя. Он принимает id пользователя и необязательный параметр время блокировки пользователя. С помощью метода myQuery, описанного в пункте 7.3.2, в базу данных передаются данные о том, что пользователь заблокирован и на какой срок он заблокирован. Сам метод представлен на рисунке 66.

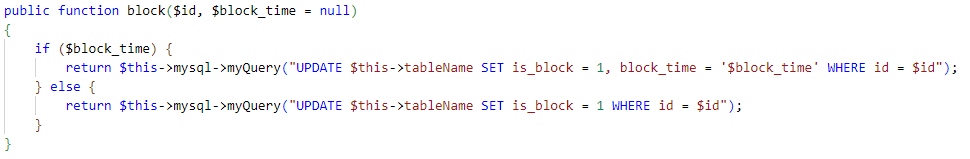
****

Рисунок 66 – метод block

7.15.2 Метод commentDelete

Этот метод реализует функцию администратора удаление комментарий. Он принимает в качестве аргумента id комментария. Внутри метода создается объект класса Comment, описанного в пункте 7.14. Туда передается объект класса Post, в него передается объект класса User, поэтому их тоже нужно создать. После этого внутри метода вызывается метод delete класса Comment, описанный в пункте 7.14.6, туда передается id комментария, возвращается результат этого метода. Сам метод commentDelete представлен на рисунке 67.

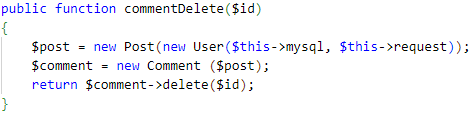
****

Рисунок 67 – метод commentDelete

7.15.3 Метод postDelete

Этот метод реализует функцию администратора удаление поста. Он принимает в качестве аргумента id поста. Внутри метода создается объект класса Comment, описанного в пункте 7.14. Туда передается объект класса Post, в него передается объект класса User, поэтому их тоже нужно создать. После этого с помощью метода myQuery, описанного в пункте 7.3.2, из базы данных возвращается массив комментариев к указанному посту. Эти комментарии удаляются с помощью метода commentDelete, описанного в пункте 7.15.2. После этого сам пост удаляется с помощью метода delete класса Post, описанного в пункте 7.10.8, возвращается его результат. Сам метод postDelete представлен на рисунке 68.

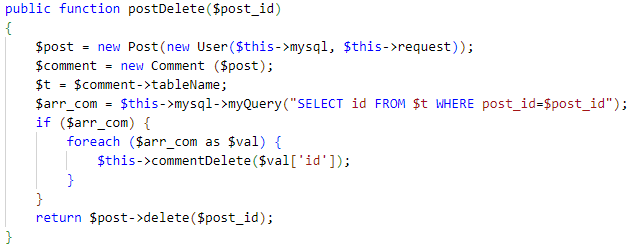
****

Рисунок 68 – метод postDelete

7.15.4 Метод userPostsDelete

Этот метод предназначен для реализации функции администратора удаление постов одного пользователя, для этого он принимает id. Внутри метода создается объект класса Post, описанный в пункте 7.10, в него передается объект класса User, поэтому его тоже нужно создать. После этого с помощью метода myQuery, описанного в пункте 7.3.2, из базы данных возвращается массив постов к указанному пользователю. Эти посты удаляются с помощью метода postDelete, описанного в пункте 7.15.3. Сам метод userPostsDelete представлен на рисунке 69.

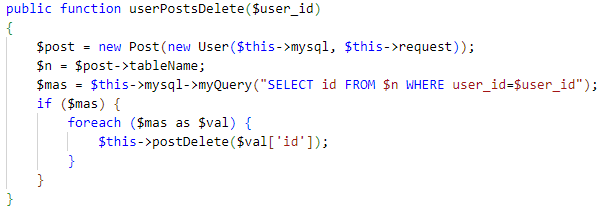
****

Рисунок 69 – метод userPostsDelete

7.16 Отображение поста на странице просмотра поста

На странице просмотра поста есть кнопки редактирования и удаления. Они отображаются вверху и внизу поста. Кнопка удаления поста отображается если пользователь администратор или если пользователь авторизован и его id совпадает с user\_id текущего поста. В её атрибуте href формируется ссылка для удаления поста с помощью метода getLink, описанного в пункте 7.6.2, туда передается ссылка на текущую страницу, токен пользователя и id поста, параметр delete со значением id поста.

Кнопка редактирования поста отображается если пользователь авторизован и его id совпадает с user\_id текущего поста. В её атрибуте href формируется ссылка с помощью метода getLink, описанного в пункте 7.6.2, туда передается ссылка на страницу редактирования поста, токен пользователя и id поста.

Для отображения поста на странице формируется HTML-код, куда добавляются атрибуты класса POST, в которых содержится логин пользователя, название поста, количество комментариев, текст поста. Дата создания поста отображается через метод changeDate, описанный в пункте 7.4.5.

Сам код отображения поста на странице просмотра представлен на рисунке 70.



Рисунок 70 – код отображения поста

При отображении поста в GET-запросе через параметр «post» передается id нужного поста. Поэтому в файле инициализации страницы просмотра поста через метод get, описанный в пункте 7.2.5, делается проверка передан ли нужный параметр. Если он передан, то по нему в атрибуты объекта класса Post загружаются данные о посте с помощью метода findOne, описанного в пункте 7.10.5. Если он не передан, то с помощью метода redirect, описанного в пункте 7.6.3, идет перенаправление на главную страницу. Этот код представлен на рисунке 71.



Рисунок 71 – код загрузки данных для отображения поста

7.17 Реализация функции удаления поста

В файле инициализации страницы просмотра поста через метод get, описанный в пункте 7.2.5, делается проверка передан ли параметр удаления поста. Если он передан и пользователь авторизован, но не администратор, происходит проверка на количество комментариев к посту. Если комментариев нет, то с помощью метода delete, описанного в пункте 7.10.8, пост удаляется, и пользователь переходит на главную страницу с помощью метода redirect, описанного в пункте 7.6.3. Если у поста есть комментарии, то в атрибут $delete класса Post добавляется сообщение об ошибке, оно отображается на странице просмотра поста. Если пользователь администратор, то вызывается метод postDelete, описанный в пункте 7.15.3, после этого пользователь переходит на главную страницу redirect, описанного в пункте 7.6.3. Сам код удаления поста представлен на рисунке 72.

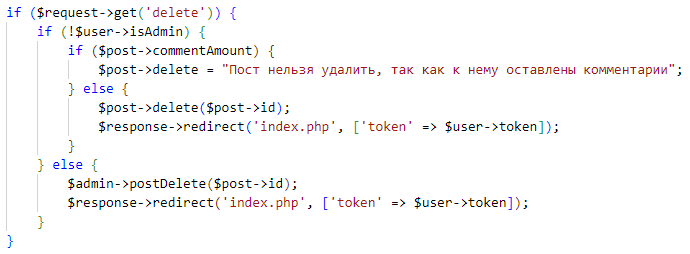


Рисунок 72 – код удаления поста

7.18 Реализация функций работы с комментариями

Список комментариев отображается на странице просмотра поста. Для получения информации о комментариях вызывается метод commentList, описанный в пункте 7.14.5, он возвращает массив. После чего для каждого отдельного элемента полученного массива формируется ссылка для удаления комментария с помощью метода getLink, описанного в пункте 7.6.2, туда передается ссылка на страницу просмотра поста, токен пользователя, id поста, параметр «commentDelete», в котором указывается id удаляемого комментария.

Для отображения одного комментария на странице формируется HTML-код, куда добавляются атрибуты класса Comment, в которых содержится имя, фамилия, отчество пользователя, текст комментария. Дата создания комментария отображается через метод changeDate, описанный в пункте 7.4.5.

Рядом с каждым комментарием есть кнопка удаления комментария. Она появляется, если пользователь является администратором, в ее атрибут href передается ранее сформированная ссылка на удаление. Сам код отображения списка комментариев на странице просмотра поста представлен на рисунке 73.

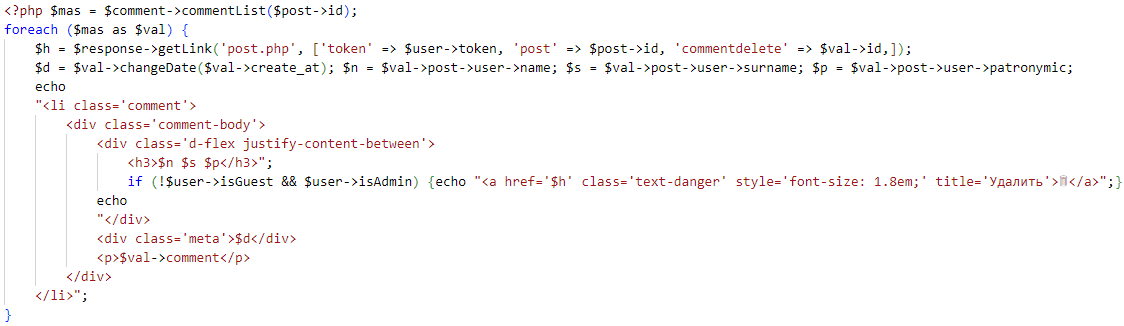


Рисунок 73 – отображение списка комментариев

После списка комментариев есть форма создания нового комментария. Она отображается только если пользователь авторизован, но не является администратором, и id пользователя не равно user\_id у поста. Для настройки формы создания комментария указывается файл для обработки формы и метод запроса для ее передачи. По умолчанию для поля формы ставится значение атрибута $comment класса Commment, в него записывается данные отправленной формы. При этом содержимое атрибута $comment отображается с помощью метода brReplace, описанного в пункте 7.4.4. В случае неверно введенных данных под полем отображается атрибут валидации $validateComment. Сама форма создания комментария представлена на рисунке 74.

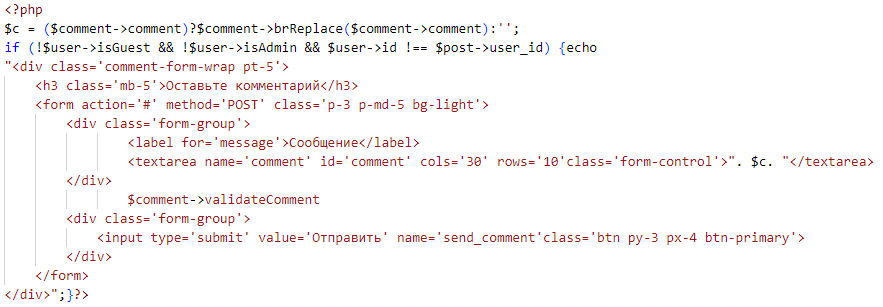


Рисунок 74 – форма создания комментария

В файле работы с комментариями производится проверка, что пользователь авторизован и при этом не является администратором, проверка на метод запроса POST с помощью класса Request, описанного в пункте 7.2. Через метод load, описанный в пункте 7.14.2 в объект класса Comment загружается значение поля создания комментария, оно проверяются через метод validate, описанный в пункте 7.14.1 Если валидация прошла успешно, то данные сохраняются в базу данных через метод save, описанный в пункте 7.14.3. После чего пользователь переходит на страницу просмотра поста с помощью метода redirect, описанного в пункте 7.6.3.

При удалении комментария в GET-запросе через параметр «commentdelete» передается id удаляемого комментария. Поэтому в файле работы с комментариями через метод get, описанный в пункте 7.2.5, делается проверка передан ли нужный параметр. Если он передан и пользователь авторизован, но не является администратором, то вызывается метод commentDelete, описанный в пункте 7.15.2. После чего пользователь переходит на страницу просмотра поста с помощью метода redirect, описанного в пункте 7.6.3. Сам файл работы с комментариями представлен на рисунке 75.



Рисунок 75 – файл работы с комментариями

7.19 Реализация отображения списка пользователей

Для отображения списка пользователей на странице создается таблица. Массив с данными о пользователях извлекается с помощью метода myQuery, описанного в пункте 7.3.2. После чего данные оттуда помещаются в поля таблицы. Отображаются поля только незаблокированных пользователей. Напротив каждого пользователя есть кнопки временной и постоянной блокировки, ссылки на них формируются с помощью метода getLink, описанного в пункте 7.6.2. Для ссылки на временную блокировку передается название страницы блокировки, токен пользователя и id блокируемого пользователя. Для ссылки на постоянную блокировку передается название текущей страницы, токен пользователя и id блокируемого пользователя. Код отображения списка пользователей представлен на рисунке 76.



Рисунок 76 – таблица с пользователями

7.20 Реализация постоянной блокировки пользователей

В файле инициализации страницы списка пользователей вызывается метод unblock, описанный в пункте 7.4.9, чтобы разблокировать пользователей, чей срок блокировки закончился. При блокировке пользователя в GET-запросе через параметр «user» передается id блокируемого пользователя. Поэтому в файле инициализации страницы списка пользователей через метод get, описанный в пункте 7.2.5, делается проверка передан ли нужный параметр. Если он передан и пользователь является администратором, то вызывается метод block, в него передается id пользователя, вызывается метод userPostsDelete, в него передается id пользователя. После этого пользователь переходит на страницу просмотра пользователей. Сам файл инициализации страницы просмотра списка пользователей представлен на рисунке 77.



Рисунок 77 – файл инициализации страницы списка пользователей

7.21 Реализация временной блокировки пользователя

При блокировке пользователя в GET-запросе через параметр «user» передается id блокируемого пользователя. Поэтому в файле инициализации страницы блокировки пользователя через метод get, описанный в пункте 7.2.5, делается проверка передан ли нужный параметр. Если он передан, то вызывается метод block, в него передается id пользователя и дата блокировки, которая преобразуется через метод sqlFormat. После этого пользователь переходит на страницу просмотра пользователей. Сам файл инициализации страницы временной блокировки пользователей представлен на рисунке 78.

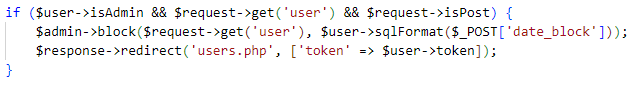


Рисунок 78 – файл инициализации временной блокировки пользователей

8 Составление Mind Map системы

На основе созданного интерфейса и функциональности системы была составлена её интеллект-карта. Она помогает при составлении тест-кейсов и отслеживании покрытия тестами функций и интерфейса системы. Интеллект-карта представлена на рисунке.

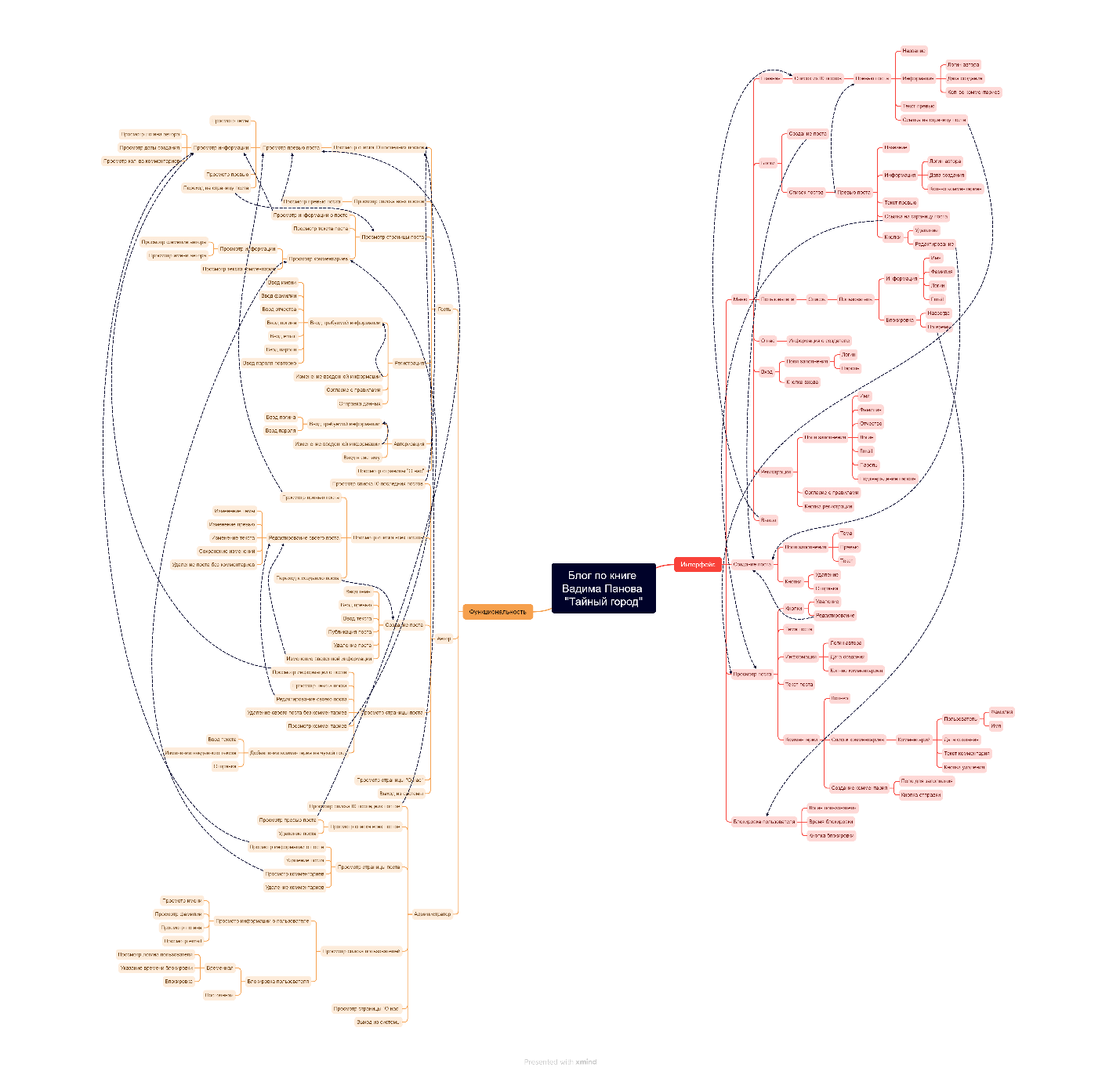
****

Рисунок - Mind Map системы

9 Составление тест-кейсов для тестирования интерфейса системы

Для тестирования системы были составлены тест-кейсы, проверяющие обработку данных, которые вводит пользователь в формы отправки данных на созданном сайте, они представлены в таблица 5 – таблица 8.

Таблица 5 – тест-кейсы для поля авторизации

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Login | Password | Ожидаемый результат | Что проверяется |
| 1 | Логин, который есть в БД | Пароль, соответствующий логину | Пользователь входит в систему, попадает на главную страницу | Авторизация |
| 2 | Логин которого нет в БД | 123456 | Сообщение «Неправильно введен логин» | Валидация логина |
| 3 | Не заполнять | 123456 | Сообщение «Логин пользователя не введен» | Валидация логина |
| 4 | Логин, который есть в БД | Не заполнять | Сообщение «Пароль пользователя не введен» | Валидация пароля |
| 5 | Логин, который есть в БД | Пароль, несоответствующий логину | Сообщение «Пароль введен неверно» | Проверка правильности пароля пользователя |
| 6 | Логин заблокированного пользователя | Пароль, соответствующий логину | Сообщение «Пользователь заблокирован» | Авторизация заблокированного пользователя |
| 7 | Пробел | 123456 | Сообщение «Логин пользователя не введен» | Очистка введенных данных |
| 8 | Логин, который есть в БД | <b></b> | Сообщение «Пароль пользователя не введен» | Очистка введенных данных |

Таблица 6 – тест-кейсы для поля регистрации

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| № | Name | Surname | Patronymic | Login | Email | Password | Password repeat | Rules | Ожидаемый результат | Что проверяется |
| 1 | Иван | Иванов | Иванович | v | v@v | 123456 | 123456 | Подтвердить | В БД добавляется новая строка, пользователь входит в систему, попадает на главную страницу | Регистрация |
| 2 | Иван | Иванов | Иванович | v | v@v | 123456 | 123456 | Не подтверждать | Сообщение «Необходимо согласиться с правилами регистрации» | Заполнения согласия с правилами регистрации |
| 3 | Не заполнять | Иванов | Иванович | v | v@v | 123456 | 123456 | Подтвердить | Сообщение «Имя пользователя не введено» | Валидация имени |
| 4 | Иван | Не заполнять | Иванович | v | v@v | 123456 | 123456 | Подтвердить | Сообщение «Фамилия пользователя не введена» | Валидация фамилии |
| 5 | Иван | Иванов | Иванович | Не заполнять | v@v | 123456 | 123456 | Подтвердить | Сообщение «Логин пользователя не введен» | Валидация логина |
| 6 | Иван | Иванов | Иванович | Логин, который уже есть в БД | v@v | 123456 | 123456 | Подтвердить | Сообщение «Пользователь с таким логином уже существует» | Валидация логина |
| 7 | Иван | Иванов | Иванович | v | Не заполнять | 123456 | 123456 | Подтвердить | Сообщение «Email пользователя не введен» | Валидация email |
| 8 | Иван | Иванов | Иванович | v | v | 123456 | 123456 | Подтвердить | Сообщение «Адрес электронной почты должен содержать символ @» | Валидация email |

Продолжение таблицы 6

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| № | Name | Surname | Patronymic | Login | Email | Password | Password repeat | Rules | Ожидаемый результат | Что проверяется |
| 9 | Иван | Иванов | Иванович | v | v@ | 123456 | 123456 | Подтвердить | Сообщение «Адрес электронной почты должен содержать символ @» | Валидация email |
| 10 | Иван | Иванов | Иванович | v | Email, который уже есть в БД | 123456 | 123456 | Подтвердить | Сообщение «Пользователь с таким email уже существует» | Валидация email |
| 11 | Иван | Иванов | Иванович | v | v@v | Не заполнять | 123456 | Подтвердить | Сообщение «Пароль пользователя не введен», «Подтверждение пароля введено неверно» | Валидация пароля |
| 12 | Иван | Иванов | Иванович | v | v@v | 12345 | 123456 | Подтвердить | Сообщение «Пароль должен быть не менее 6 символов», «Подтверждение пароля введено неверно» | Валидация пароля |
| 13 | Иван | Иванов | Иванович | v | v@v | 123456 | Не заполнять | Подтвердить | Сообщение «Подтвердите введенный пароль» | Валидация пароля |
| 14 | Пробел | Иванов | Иванович | v | v@v | 123456 | 123456 | Подтвердить | Сообщение «Имя пользователя не введено» | Очистка введенных данных |
| 15 | <b> | Иванов | Иванович | v | v@v | 123456 | 123456 | Подтвердить | Сообщение «Имя пользователя не введено» | Очистка введенных данных |

Таблица 7 – тест-кейсы для поля создания/редактирования поста

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Post title | Post preview | Post content | Ожидаемый результат | Что проверяется |
| 1 | тема | превью | Проверка  <b>очистки</b>  тегов | Пост создан, пользователь переходит на страницу просмотра созданного поста. В БД в таблице «post» в строке с новым постом в графе «content» запись «Проверка<br>очистки<br>тегов» | Очистка введенных данных, создание поста |
| 2 | тема | превью | Пробел | Сообщение «Текст поста не введен» | Очистка введенных данных |
| 3 | Не заполнять | превью | текст | Сообщение «Тема поста не введена» | Валидация темы |
| 4 | тема | Не заполнять | текст | Сообщение «Превью поста не введено» | Валидация превью |
| 5 | тема | превью | Не заполнять | Сообщение «Текст поста не введен» | Валидация текста |
| 6 | тема | превью | Пробел | Сообщение «Текст поста не введен» | Очистка введенных данных |

Таблица 8 – тест-кейсы для поля отправки комментариев

|  |  |  |  |
| --- | --- | --- | --- |
| № | Comment | Ожидаемый результат | Что проверяется |
| 1 | Короткий  <b>комментарий</b> | Комментарий создан и отображается на странице просмотра поста. В БД в таблице «comment» в строке с новым комментарием в графе «comment» запись «Короткий<br>комментарий» | Очистка введенных данных, создание поста |
| 2 | Пробел | Сообщение «Текст комментария не введен» | Очистка введенных данных |
| 3 | Не заполнять | Сообщение «Текст комментария не введен» | Валидация текста комментария |

ЗАКЛЮЧЕНИЕ

Таким образом, за время прохождения учебной практики были освоены все необходимые практические компетенции (ПК 5.1 — ПК 5.7), предполагаемые профессиональным модулем ПМ.05: «Проектирование и разработка информационных систем». В ходе работы мною были приобретены и отточены следующие навыки:

- разработка технического задания на веб-приложение в соответствии с требованиями;

- разработка проектной документации на разработку информационной системы в соответствии с требованиями;

- разработка подсистемы безопасности информационной системы в соответствии с техническим заданием;

- разработка модулей информационной системы в соответствии с техническим заданием;

- тестирование информационной системы на этапе опытной эксплуатации с фиксацией выявленных ошибок кодирования в разрабатываемых модулях информационной системы;

- разработка технической документации на эксплуатацию информационной системы;

- оценка информационной системы для выявления возможности ее модернизации.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Моделирование на UML [Электронный ресурс]/ Денис Иванов – 09.04.2013. – URL: <http://book.uml3.ru/content> Дата обращения: 19.05.2023.

2. Презентация на тему "Нотация IDEF0" [Электронный ресурс]/ИНФОУРОК — URL: https://infourok.ru/prezentaciya-na-temu-notaciya-idef0-4239984.html Дата обращения: 15.05.2023.

3. DFD (Data Flow Diagram) Диаграммы — зачем они нужны и какие бывают [Электронный ресурс]/ Даниил Джепаров//Хабр – 30.05.2022. – URL: <https://habr.com/ru/articles/668684/> Дата обращения: 16.05.2023.

4. PostgreSQL: Документация: 15: Часть II. Язык SQL [Электронный ресурс]/PostgresPro – URL: <https://postgrespro.ru/docs/postgresql/15/sql> Дата обращения: 27.05.2023.

5. MySQL 8.0 Reference Manual [Электронный ресурс]/MySQL – URL: <https://dev.mysql.com/doc/refman/8.0/en/> Дата обращения: 27.05.2023.

6. MySQL 8.0 Reference Manual [Электронный ресурс]/MySQL – URL: <https://dev.mysql.com/doc/refman/8.0/en/> Дата обращения: 27.05.2023.

7. Руководство по PHP [Электронный ресурс]/PHP – URL: https://www.php.net/manual/ru/ Дата обращения: 27.05.2023.

8. Перевод стандартов PSR-0, PSR-1, PSR-2, PSR-3, PSR-4 [Электронный ресурс]/ Хроники детерминированности – 15.09.2014 – URL: <https://svyatoslav.biz/misc/psr_translation/> Дата обращения: 27.05.2023.

9. Савин Р. Тестирование Дот Ком, или Пособие по жестокому обращению с багами в интернет-стартапах [Текст]/ Савин Р. – М.: Дело, 2007. – 312 с.