ПРАВИТЕЛЬСТВО САНКТ-ПЕТЕРБУРГА

КОМИТЕТ ПО ОБРАЗОВАНИЮ

Санкт-Петербургское государственное бюджетное

профессиональное образовательное учреждение

«Радиотехнический колледж»

ОТЧЕТ

По учебной практике

Специальность: 09.02.07 «Информационные системы и программирование»

Квалификация: «Разработчик веб и мультимедийных приложений»

Выполнил:

Студент 2 курса

гр. ИВ1-21

Тобольшина Анастасия Алексеевна

Преподаватель:

Харин Евгений Николаевич

Санкт-Петербург 2023 г.

# СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_3hhj8t79vazi)

[1 USE-CASE ДИАГРАММА 4](#_ykam1up02f5)

[1.1 Выделенные актёры: 4](#_7e1cug5cfofr)

[1.2 Выделенные прецеденты для актёров: 5](#_4ll4dpr8d5fn)

[2 IDEF0 ДИАГРАММА 7](#_p42uaab0eopg)

[2.1 Сущности, которые взаимодействуют с системой: 7](#_b9ti18r12lan)

[2.2 Потоки управления, контролирующие действия на сайте: 7](#_nrqrrccdsfv2)

[2.3 Входные потоки, обрабатываемые сайтом: 8](#_6lu048hhbq2l)

[3 ДЕКОМПОЗИЦИЯ IDEF0 ПЕРВОГО УРОВНЯ 9](#_lg433mluxo3t)

[3.1 Блок “Работа с пользователем”: 9](#_qiq7hkstrzeg)

[3.2 Блок “Работа с постами”: 10](#_s3alhyg4qdwm)

[3.3 Блок “Работа с комментариями”: 10](#_2gnpk3nt521k)

[3.4 Блок “Администрирование”: 10](#_bd0e2ifd7kb4)

[4 ER ДИАГРАММА БАЗЫ ДАННЫХ 12](#_lmbhn8jg71sw)

[4.1 Атрибуты таблицы “users”: 12](#_y2pq0nhb3wsi)

[4.2 Атрибуты таблицы “roles”: 13](#_qppfgkwvz29r)

[4.3 Атрибуты таблицы “posts”: 13](#_uwk2txsg9vu1)

[4.4 Атрибуты таблицы “comments”: 13](#_ulpmom7q9m42)

[5 РЕАЛИЗАЦИЯ ИНФОРМАЦИОННОЙ СИСТЕМЫ 14](#_2eo0ui2jjjhz)

[5.1 Подготовка к реализации функционала системы. Часть 1 14](#_st3lwe61zp3s)

[5.2 Подготовка к реализации функционала системы. Часть 2 16](#_jh6bicnh4czy)

[5.3 Реализация регистрации пользователя 18](#_bs0x6vr9nzdw)

[5.4 Реализация авторизации пользователя 20](#_hzyb0ggxkjmw)

[6 MIND MAP СИСТЕМА 23](#_rcy1n934kgtl)

[ЗАКЛЮЧЕНИЕ 24](#_tobcg9ea059h)

# ВВЕДЕНИЕ

В данном отчете будет рассмотрено выполнение мной учебной практики, в рамках которой была разработана информационная система для создания, просмотра и редактирования постов. Целью данной работы являлось изучение основных принципов web-разработки, а также получение опыта в создании пользовательских интерфейсов и баз данных.

В работе было рассмотрено несколько этапов, начиная с проектирования системы до ее развертывания на сервере.

В результате работы была создана функциональная информационная система. В данном отчете будут рассмотрены основные характеристики системы, ее особенности и преимущества для пользователя.

# 1 USE-CASE ДИАГРАММА

То, каким образом пользователь может взаимодействовать с информационной системой, видно на рисунке 1.

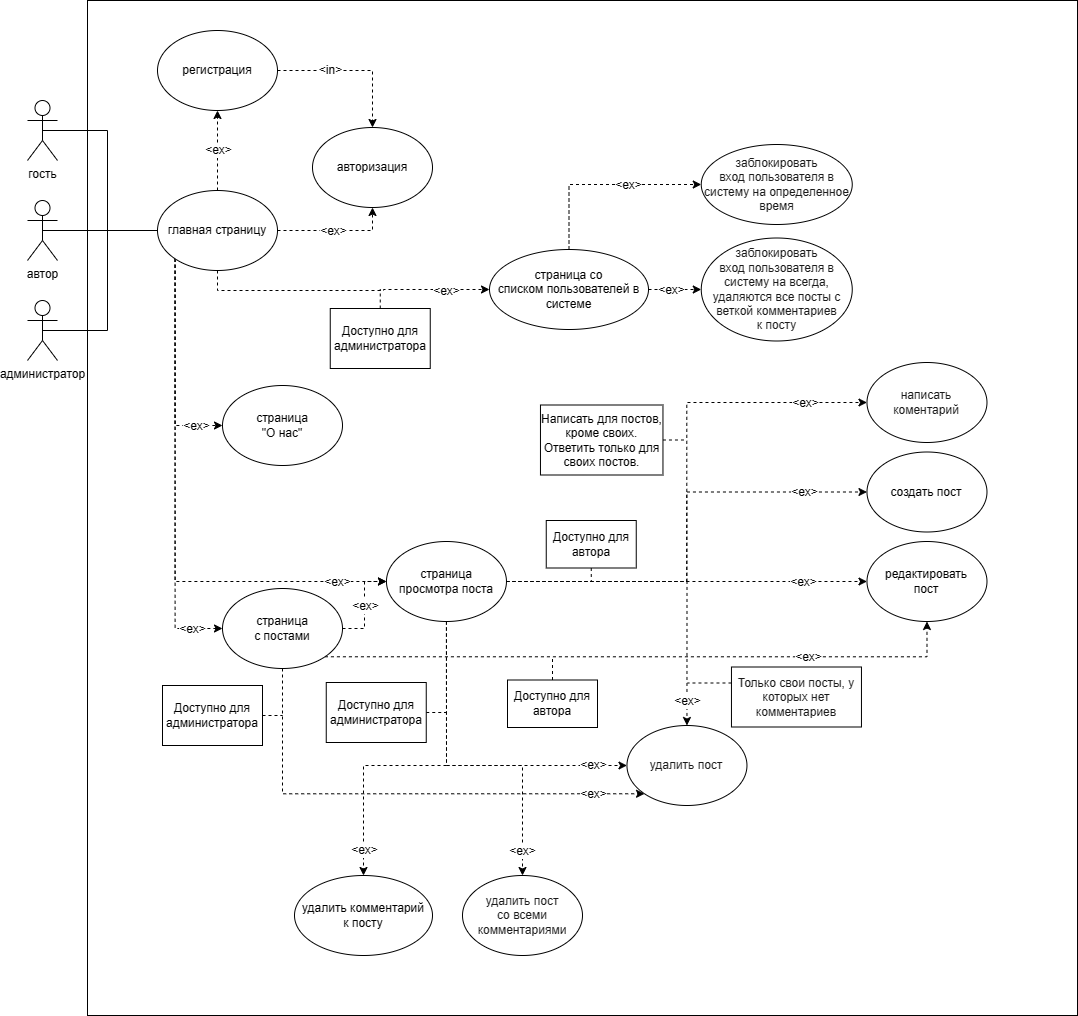


Рисунок 1 - Use-Case диаграмма

## 1.1 Выделенные актёры:

* Гость, неавторизованный пользователь,
* Автор, авторизованный пользователь,
* Администратор, авторизованный пользователь, с правами управления системы.

## 1.2 Выделенные прецеденты для актёров:

* Главная страница:

1. Доступна лента с десятью постами;
2. Позволяет перейти на страницу регистрации;
3. Позволяет перейти на страницу авторизации;
4. Позволяет перейти на страницу “О нас”;
5. Позволяет перейти на страницу с постами;
6. Позволяет перейти на страницу просмотра поста.

* Регистрация:

1. Позволяет зарегистрироваться на сайте;
2. После регистрации переходит на страницу авторизации.

* Авторизация:

1. Позволяет зарегистрированному пользователю авторизоваться в системе;
2. После авторизации переходит на главную страницу.

* Страница “О нас”

1. Представлена информация по описанию системы;
2. Представлена информация о разработчике.

* Страница с постами:

1. Страница содержит ленту превью всех постов, с указанием даты и времени создания поста, название (тема) поста, автора (логина) и анонса (превью) поста;
2. Для автора поста доступно редактирование поста;
3. Для администратора доступно удаление поста.

* Страница просмотра поста:

1. Содержит текст поста;
2. Содержит ветку комментариев к посту;
3. Содержит форму создания комментария к посту с учетом условий комментирования поста;
4. Для автора позволяет перейти на страницу редактирования поста;
5. Для автора и администратора доступно удаление поста;
6. Для автора доступно удаление комментария.

* Страница со списком пользователей в системе:

1. Позволяет заблокировать вход пользователя в систему на определенный срок;
2. Позволяет заблокировать вход пользователя в систему навсегда.

* Редактировать пост (позволяет автору редактировать пост),
* Создать пост (позволяет автору создать пост),
* Написать комментарий (позволяет автору написать комментарий с учетом условий комментирования поста).

# 2 IDEF0 ДИАГРАММА

Процесс создания поста, который можно реализовать посредством информационной системы, изображен на рисунке 2.

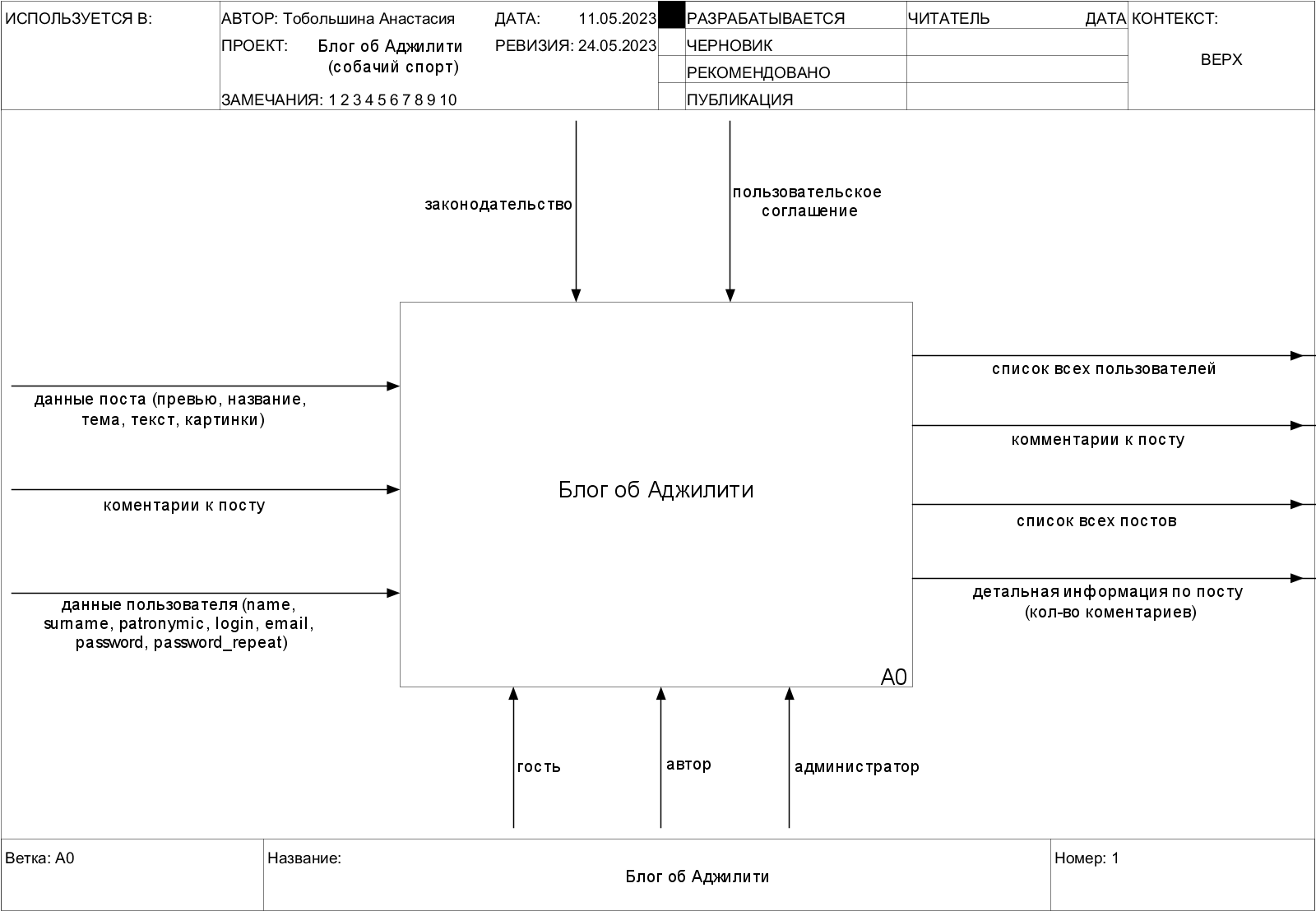


Рисунок 2 - IDEF0 диаграмма

Функциональным блоком первого уровня IDEF диаграммы является сам сайт.

## 2.1 Сущности, которые взаимодействуют с системой:

* Гость,
* Автор, авторизованный пользователь,
* Администратор, авторизованный пользователь.

## 2.2 Потоки управления, контролирующие действия на сайте:

* Пользовательское соглашение,
* Законодательство.

## 2.3 Входные потоки, обрабатываемые сайтом:

* Данные пользователя (name, surname, patronymic, login, email, password, password\_repeat).
* Комментарии к посту. На выходе преобразуются в ветку комментариев;
* Данные поста (превью, название, тема, текст, картинки). На выходе преобразуются в пост, выложенный автором;

Данные для комментария (текст комментария, данные о пользователе (авторе комментария)) преобразовываются на выходе в комментарий к посту.

# 3 ДЕКОМПОЗИЦИЯ IDEF0 ПЕРВОГО УРОВНЯ

На рисунке 3 мы можем детально рассмотреть процесс информационной системы.

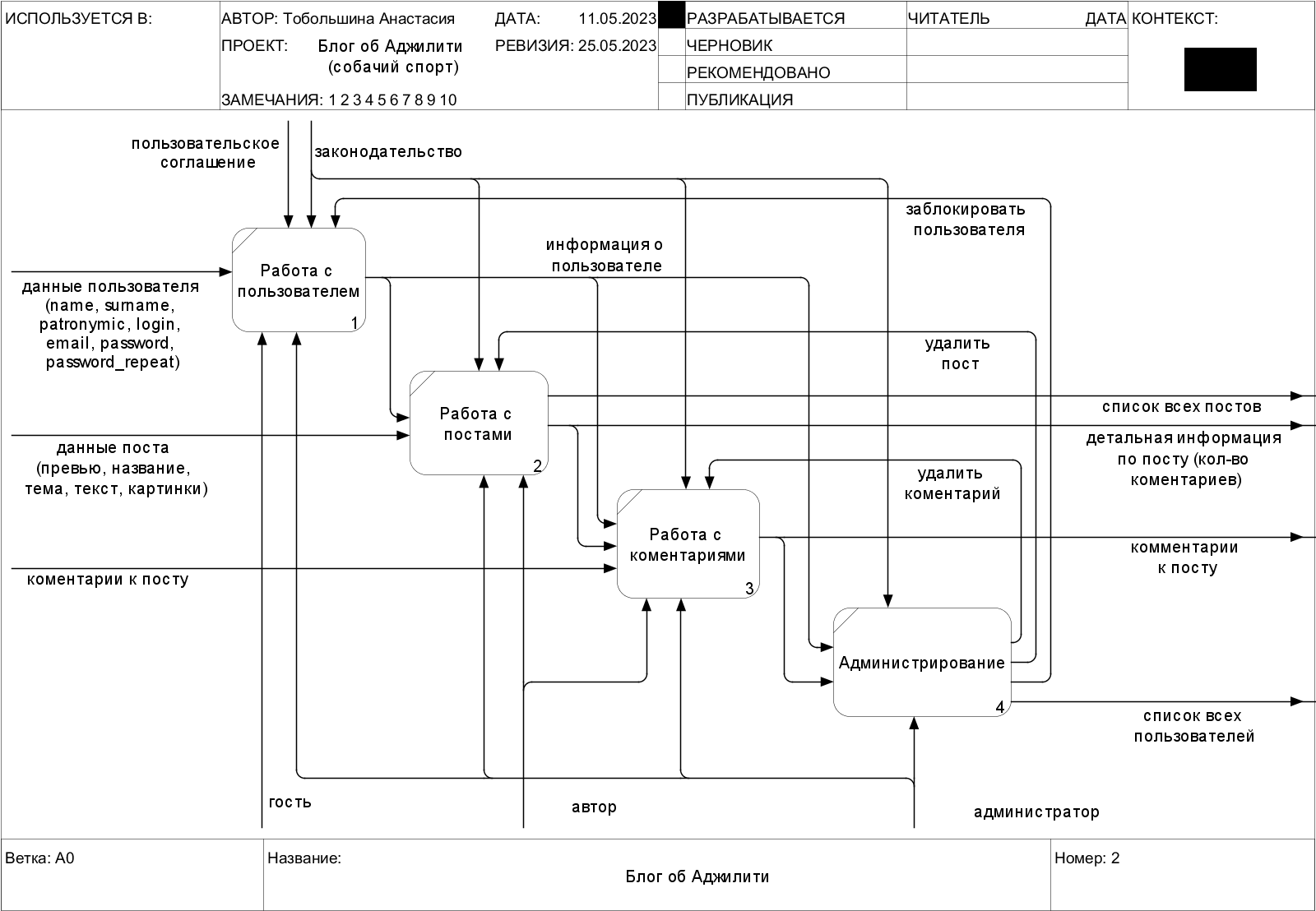


Рисунок 3 - Декомпозиция диаграммы IDEF0 первого уровня

## 3.1 Блок “Работа с пользователем”:

* Входные потоки, обрабатываемые блоком:

1. Данные пользователя (name, surname, patronymic, login, email, password, password\_repeat).

* Сущности, которые взаимодействуют с системой:

1. Гость;
2. Администратор.

* Потоки управления, контролирующие действия:

1. Пользовательское соглашение;
2. Законодательство.

* Потоки на выходе:

1. Информация о пользователе.

## 3.2 Блок “Работа с постами”:

* Входные потоки, обрабатываемые блоком:

1. Данные поста (превью, название, тема, текст, картинки).

* Сущности, которые взаимодействуют с системой:

1. Автор;
2. Администратор.

* Потоки управления, контролирующие действия:

1. Законодательство;
2. Администрирование.

* Поток на выходе:

1. Детальная информация по посту.

## 3.3 Блок “Работа с комментариями”:

* Входные потоки, обрабатываемые блоком:

1. Комментарии к посту.

* Сущности, которые взаимодействуют с системой:

1. Автор;
2. Администратор.

* Потоки управления, контролирующие действия:

1. Законодательство;
2. Администрирование.

* Поток на выходе:

1. Комментарий к посту.

## 3.4 Блок “Администрирование”:

* Входные потоки, обрабатываемые блоком:

1. Информация о пользователе.

* Сущности, которые взаимодействуют с системой:

1. Администратор.

* Потоки управления, контролирующие действия:

1. Законодательство.

* Потоки на выходе:

1. Список всех пользователей.

Необходима таблица: поле, тип данных, ограничения

# 4 ER ДИАГРАММА БАЗЫ ДАННЫХ

На рисунке 1 показана база данных сайта.

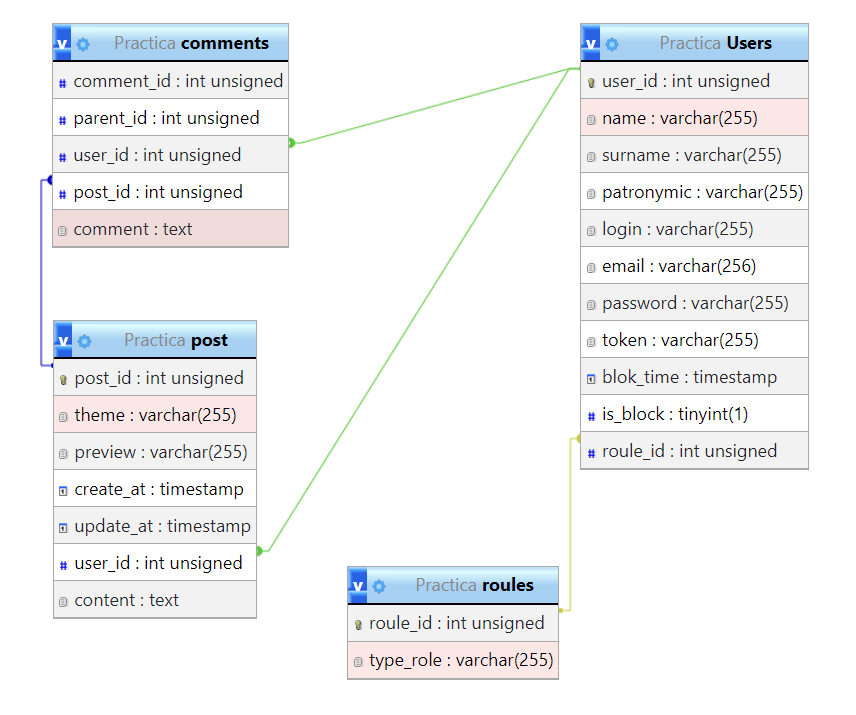


Рисунок 4 - ER диаграмма

На ER диаграмме представлено несколько сущностей: users (пользователи), roles (роли), posts (посты), comments (комментарии).

## 4.1 Атрибуты таблицы “users”:

* user\_id – уникальный идентификатор пользователя;
* name – имя, обязательное поле;
* surname – фамилия, обязательное поле;
* patronymic – отчество, необязательное поле;
* login – уникальный логин пользователя, обязательное поле;
* email – уникальная почта, обязательное поле;
* password – пароль пользователя;
* token – идентификатор сессии, необязательное поле;
* block\_time – время блокировки юзера, не обязательное поле;
* is\_block – статус наличия блокировки, необязательное поле;
* role\_id – идентификатор роли пользователя, обязательное поле, наследуется от roles (role\_id).

## 4.2 Атрибуты таблицы “roles”:

* role\_id – идентификатор роли пользователя;
* type\_role – наименование роли.

## 4.3 Атрибуты таблицы “posts”:

* post\_id – уникальный идентификатор поста;
* theme – тема поста, обязательное поле;
* preview – превью поста, обязательное поле;
* create\_at – дата создания поста, обязательное поле;
* update\_at – дата редактирования поста, необязательное поле;
* user\_id – идентификатор пользователя, автора поста, наследуется от users (user\_id), обязательное поле;
* content – текст поста, обязательное поле;
* picture – изображение, прикрепленное к посту, необязательное поле.

## 4.4 Атрибуты таблицы “comments”:

* comment\_id – уникальный идентификатор комментария;
* parent\_id – идентификатор комментария, на который был написан; ответ, отсылается к comments (comment\_id), необязательное поле;
* post\_id – идентификатор поста, к которому был написан комментарий, ссылается на posts (post\_id), обязательное поле;
* comment – текст комментария.

# 5 РЕАЛИЗАЦИЯ ИНФОРМАЦИОННОЙ СИСТЕМЫ

## 5.1 Подготовка к реализации функционала системы. Часть 1

Необходимо было проанализировать верстку страниц и группировать код по определенному признаку:

* в секции header – метаданные, файлы css, шрифты;
* в секции где располагается боковое меню – код бокового меню;
* код относящийся к прелоадеру;
* js скрипты.

Код переместить в отдельные файлы. С помощью метода file\_get\_contents, который показан на рисунке 5. Изменить формат файлов с html на php и подключить код к страницам, что показано на рисунке 6.

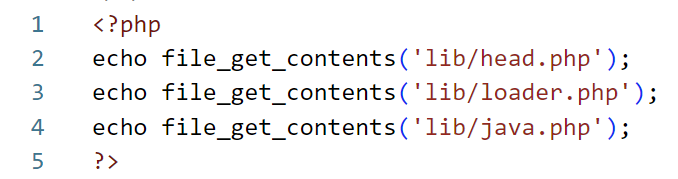


Рисунок 5 - Подключение повторяющихся фрагментов кода

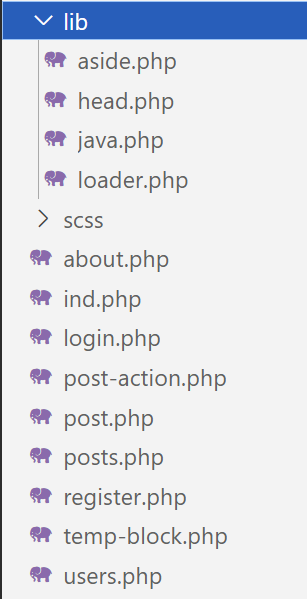


Рисунок 6 - Файлы страниц

В файле конфигурации, показанный на рисунке 7, был создан массив с пунктами меню.

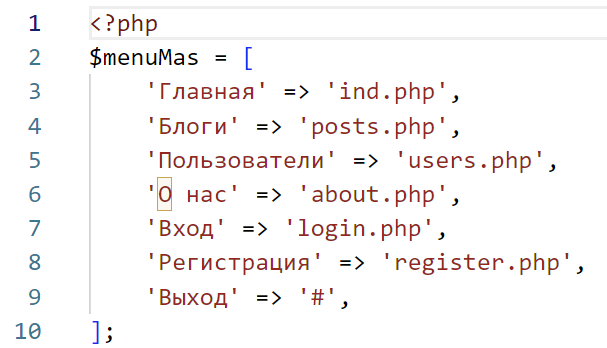


Рисунок 7 - Массив с пунктами меню

На рисунке 8 массив передается в конструктор класса Menu. Каждый элемент массива содержит ссылку на страницу и ее наименование.



Рисунок 8 - Класс Menu

Подключен файл autoloader, рисунок 9, в файле инициализации, рисунок 10, и создан экземпляр класса Menu.

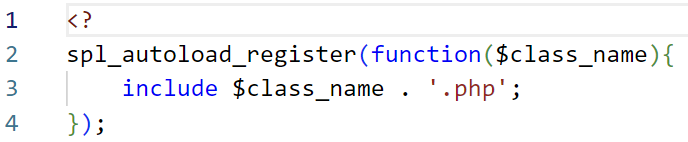


Рисунок 9 - Файл autoloader

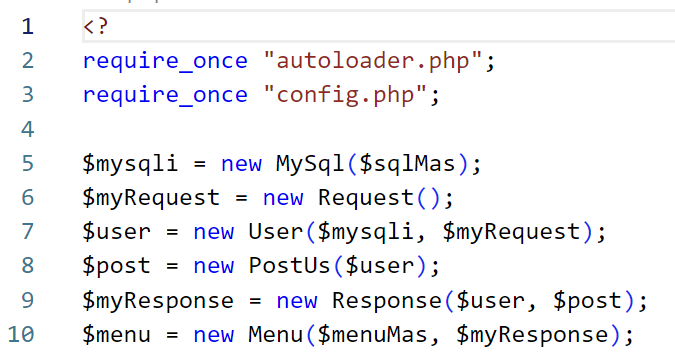


Рисунок 10 - Файл инициализации

Далее создается файл инициализации для каждой страницы, показанные на рисунке 11. На рисунке 12 каждый файл инициализации подключается с помощью функции require\_once.

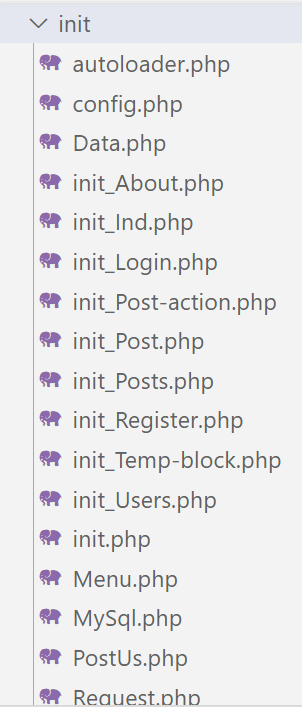


Рисунок 11 - Файлы инициализации

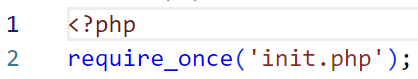


Рисунок 12 - Функция require\_once

## 5.2 Подготовка к реализации функционала системы. Часть 2

В соответствии с рисунком 13 был создан класс Request для обработки запросов.

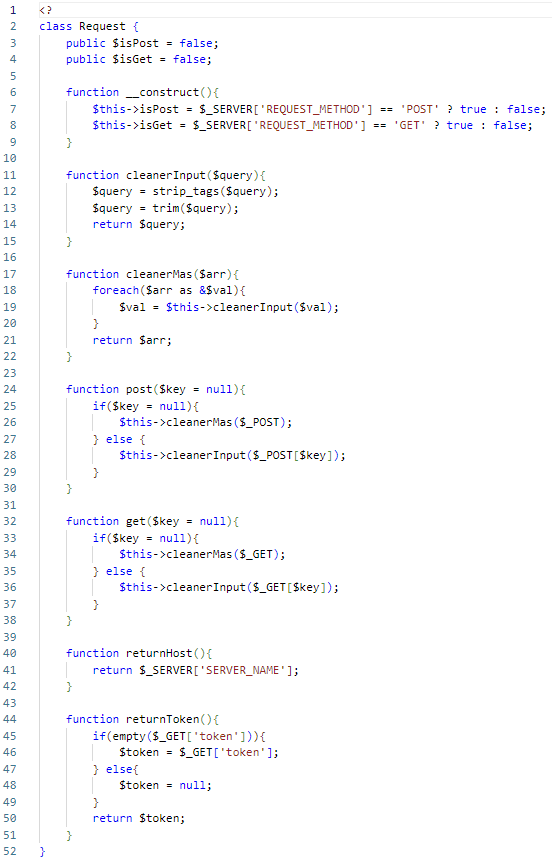


Рисунок 13 - Класс Request

В конструкторе класса были проинициализированы атрибуты. В классе были реализованы атрибуты isPost (признак, что запрос был совершен методом POST) и isGet (признак, что запрос был совершен методом GET).

Для заполнения атрибутов использовалось $\_SERVER[“REQUEST\_METHOD”]. Был создан метод очистки входного параметра, метод очистки массива, метод post, метод get, метод возвращающий host запроса из массива $\_SERVER, метод возвращающий параметр token из строки запроса.

В файле инициализации приложения создан экземпляр класса Request.

## 5.3 Реализация регистрации пользователя

Создан класс User, показанный на рисунке 14, реализующий работу с пользователем.

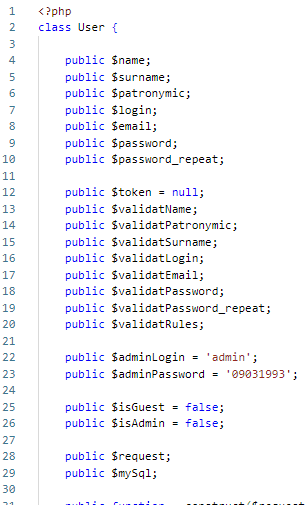


Рисунок 14 - Класс User

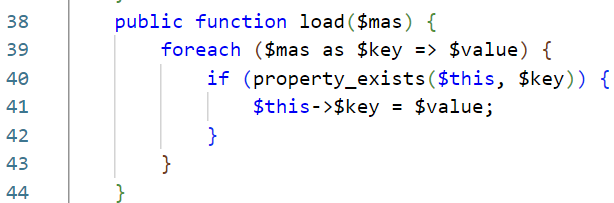
Был создан метод load, в соответствии с рисунком 15. Метод принимает массив с данными пользователя и, используя функцию property\_exists, производит загрузку данных из массива в соответствующий атрибут класса.

Рисунок 15 - Метод load

Создан метод validateRegister, показанный на рисунке 16. Метод возвращает признак сработало или нет хотя-бы одно из правил валидации. Для реализации метода использовалась функция get\_object\_vars.

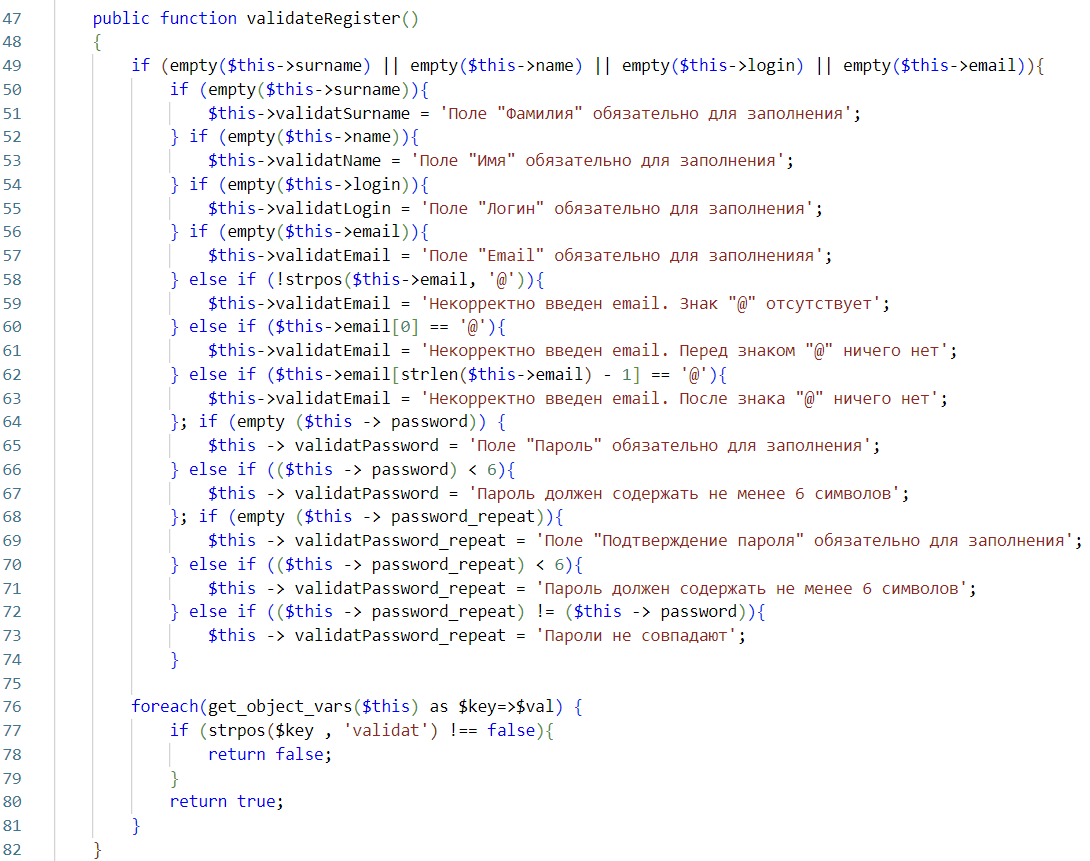


Рисунок 16 - Метод validateRegister

Также был создан метод save, показанный на рисунке 17. Этот метод сохраняет данные пользователя в базу данных, в соответствии с рисунком 18. После регистрации пользователя перекидывает на главную страницу.

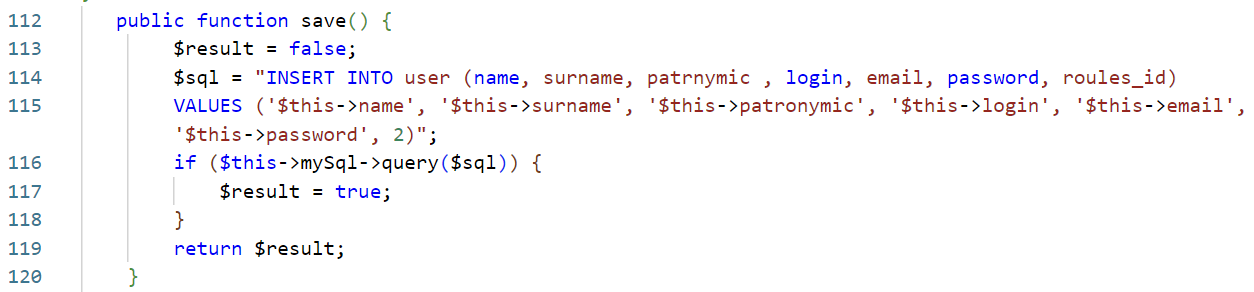


Рисунок 17 - Метод save

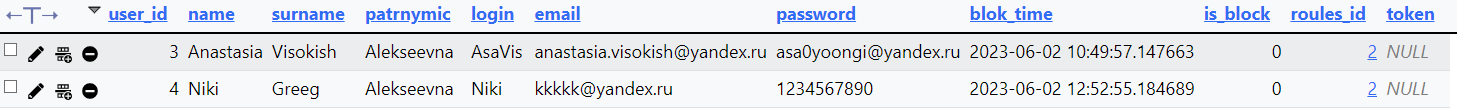


Рисунок 18 - База данных

## 5.4 Реализация авторизации пользователя

Создан метод validateLogin для валидации данных при авторизации пользователя, показанный на рисунке 19. Метод возвращает признак сработало или нет хотя-бы одно из правил валидации. Для реализации метода использовалась функция get\_object\_vars.



Рисунок 19 - Метод validateLogin

Создан метод login, показанный на рисунке 20. В методе производится проверка пароля и логина введенных пользователем и из БД. Также, используя метод load, производится загрузка данных полученных из бд в атрибуты класса. Для установки роли используется метод isAdmin. В соответствии с рисунком 21, метод isAdmin возвращает признак совпадения логина и пароля текущего пользователя с данными администратора. Сформирован токен для пользователя из набора случайных символов и записан в БД. Метод load возвращает признак успешной операции, либо возвращает информацию о некорректности данных.

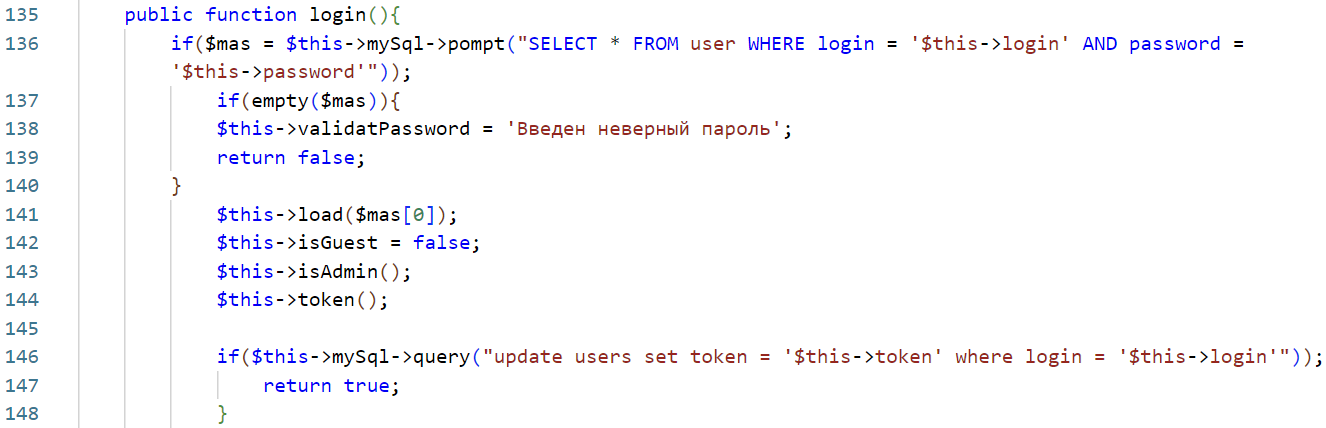


Рисунок 20 - Метод load

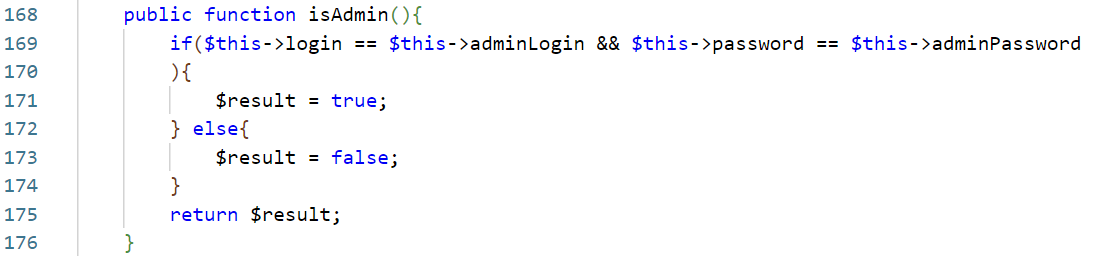


Рисунок 21 - Метод isAdmin

Создан метод identity, показанный на рисунке 22, для идентификации пользователя. Метод принимает id пользователя. Если id не передан, то данные по пользователю загружаются на основе токена.

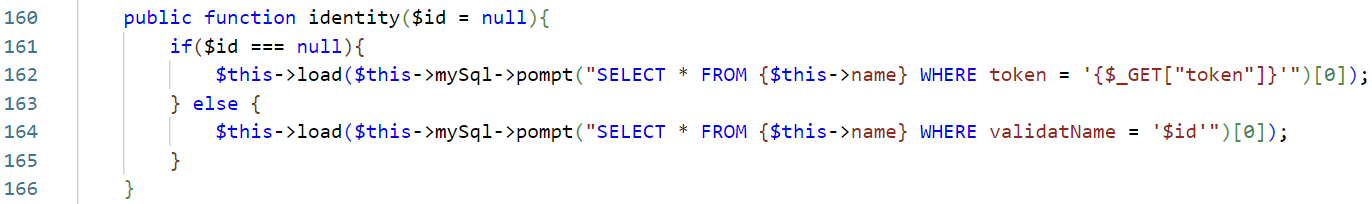


Рисунок 22 - Метод identity

Создан метод logout, показанный на рисунке 23, для «разлогирования» пользователя. Если пользователь авторизован в системе, то производится выход из системы, удалив токен в БД.

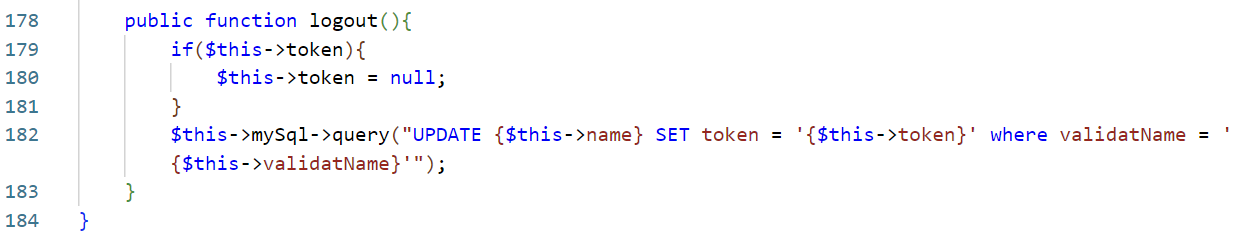


Рисунок 23 - Метод logout

# 6 MIND MAP СИСТЕМА

На данной системе на рисунке 24 наглядно показано из чего состоит информационная система.



Рисунок 24 - Mind Map система

# ЗАКЛЮЧЕНИЕ

В ходе выполнения данной учебной практики я получила необходимые знания и опыт в области web-разработки, создания пользовательских интерфейсов и баз данных.

В ходе создания информационной системы для создания, просмотра и редактирования постов были использованы современные технологии и подходы, что позволило создать функциональную, удобную и быструю систему.

Я научилась работать с такими инструментами, как PHP и SQL.

В результате выполнения учебной практики я получила опыт создания информационной системы, узнала как происходит разработка программного обеспечения с нуля до готового продукта, а также нарисовала для себя путь для дальнейшего саморазвития.