**Раздел 1. Обзорно-постановачная часть**

**1.1 анализ предметой области**

**1.2**

**обоснование использования бд**

База данных является одной из главных частей разрабатываемого веб- приложения. Главными целями проектирования базы данных является:

* Обеспечение защиты и целостности данных;
* Интерпретация и выполнение запросов;
* Управление пользователем и распределение прав доступа;
* Управление механизмом работы транзакции (операция или блокировка операции, которые совершаются вместе).

При выборе СУБД необходимо руководствоваться критериями, благодаря которым можно обеспечить достаточно хорошее взаимодействие с базами данных. В данном случае, следует выделить следующие критерии:

* Защищенность СУБД;
* Многопользовательность;
* Её высокая производительность при работе с данными;
* Возможность бесплатного доступа;
* Простота в использовании.

Для проекта была выбранна СУБД — **SurrealDB**, одна из новых баз данных разработанных для облачных сервисов и сайтов, соеденяющий в себе множество парадигм хранения данных, стремительно развивающаяся на данный момент. Имеет в себе отличия от стандартной модели IDEF0 и IDEF0х от чего снизу будет описания с целью добавления понимания кое будет видно во второй части в модели бд

**1.3**

**выбор языка програмирования**

Разработка веб-приложения осуществляется с использованием определенного набора программных средств, наиболее комфортная и стабильная работа с которыми достигается при корректном выборе основного языка разработки серверной части веб-приложения.

Наиболее подходящий язык программирования для разработки движка веб-приложения и его основного функционала определяется возможностью быстрой работой с выбранной СУБД, удобством развертывания приложения на конечный сервер

В рамках данного проекта необходимо выбрать такой язык программирования, который бы позволил поддерживать веб-приложения на протяжение долгого времени, имеет возможности «модульной» разработки и достаточно масштабируем для работы с потенциально большим количеством одновременно подключенных клиентов.

1. PHP

Это серверный язык программирования, применяемый для создания динамических веб-страниц и статичных сайтов. Он является очень популярным благодаря наличию большого количества встроенных средств и модулей для разработки различных приложений. Имеет открытый исходный код и собственную лицензию.

# 2. ПРАКТИЧЕСКАЯ ЧАСТЬ

2.1 Проектирование и разработка базы данных

ER-модель (от англ. Entity-Relationship model, модель «сущность — связь») — модель данных, позволяющая описывать концептуальные схемы предметной области.

Во время проектирования баз данных происходит преобразование схемы, созданной на основе ER-модели, в конкретную схему базы данных на основе выбранной модели данных

Сущность определяется как некоторый объект рассматриваемой предметной области, информация о котором должна быть отражена в базе данных. Этот объект должен иметь экземпляры - конкретные представители данной сущности, отличающиеся друг от друга и допускающие однозначную идентификацию.

Таблица 2.1. Описание множества сущностей.

|  |  |  |  |
| --- | --- | --- | --- |
| № | имя | Определение | Описание |
| Е1 | пользаватель | Зарагестрированный пользаватель системы включающий в себя роль и действия на сайте | Новый экземпляр появляеться после регистрации почты и пароля на сайте |
| Е2 | оценки | Содержит записи о том что люди проголосовали насчет любых предметов выставленных на сайте для продажи | Создаеться экземпляр при голосовании пользователся на предмете или др пользователе |
| Е3 | предметы | Содержит информацию о том что это за предмет ее картинку описание цену и другую информацию | Создаеться экземпляр пользователем при добавлении предмета на продажу |
| Е4 | заказы | Описывает что пользаватель покупал на сайте в виде истории покупок | Создаеться экземпляр каждый раз когда совершаеться покупка пользавателем |
| Е5 | токен | Техническая сущность содержащая токены для доступа к сайту или информацию для каких либо действий | Новый экземпляр создаеться каждый раз сайтом для клиентов сервера при авторизации |

Концептуальная модель также отражает связи между сущностями.

Таблица 2.2. Описание связей

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Пользаватеь Е1 | Оценки Е2 | Предметы Е3 | Заказы Е4 | Токен Е5 |
| Пользаватеь Е1 |  | R1 | R4 | R5 | R3 |
| Оценки Е2 | R1 |  | R2 |  |  |
| Предметы Е3 | R4 | R2 |  | R6 |  |
| Заказы Е4 | R5 |  | R6 |  |  |
| Токен Е5 | R3 |  |  |  |  |

Таблица 2.3. Описание связей

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | тип | Род сущ | Доч сущ | имя | описание |
| R1 | 1:М | Пользаватель Е1 | Оценки Е2 | Оценка пользавателя |  |
| R2 | 1:М | Предметы Е3 | Оценки Е2 | оценка на предмет |  |
| R3 | M:1 | Токены Е3 | Пользаватель Е1 | Токен авторизации |  |
| R4 | 1:1 | Предметы Е3 | Пользаватель Е1 | автор |  |
| R5 | М:1 | Заказы Е4 | Пользаватели Е1 | покупатель |  |
| R6 | 1:М | Заказы Е4 | Предметы Е3 | покупки |  |

Таблица 2.4. «users»

|  |  |  |
| --- | --- | --- |
| название | Тип данных | ключ |
| email | string |  |
| id | UUID | первичный |
| Password | string |  |
| nick | string |  |
| description | string |  |
| Индекс email | index\_uni |  |
| Индекс nick | index\_uni |  |

Таблица 2.5. «items»

|  |  |  |
| --- | --- | --- |
| название | Тип данных | ключ |
| value | int |  |
| author | record |  |
| name | string |  |
| description | string |  |
| tags | Array[].\*string |  |
| id | UUID | первичный |

Таблица 2.6. «votes»

|  |  |  |
| --- | --- | --- |
| название | Тип данных | ключ |
| id | UUID | первичный |
| item | record |  |
| user | record |  |
| value | int |  |
| comment | string |  |
| Voted | Index unique |  |

Таблица 2.8. «tokens»

|  |  |  |
| --- | --- | --- |
| название | Тип данных | ключ |
| id | UUID | первичный |
| time | datetime |  |
| user | Record |  |
| endtime | daytime |  |

Таблица 2.8. «customers»

|  |  |  |
| --- | --- | --- |
| название | Тип данных | ключ |
| id | UUID | первичный |
| customer | record |  |
| adress | string |  |
| executor | record |  |
| regtime | datetime |  |
| items | Array[].\*string |  |

UUID — автомотически генерируемый случайный первичный ключ пример item:n01ev28741y1d9a так же возможно в него написать любое другое удобное значение вместо автомотически генерируемого

Record — тип данных обозначающий ссылку на другие записи в базе данных используеться для односторонних ссылок внути бд

array[] - тип данных позволяющий хранить массив внутри записи который в свою очередь может хранить что угодно

.\*string — любой тип данных в бд можно переопределить чтоб структура данных была сохраннена в другом формате, результат при этом всегда будет один и тот же при выведении через запросы

index — тип данных хранящий все записи указанные в индексе возможно по нему ограничить записи в бд тегом unique если запись уже существует внутри индекса

так же существует полная квери для создания бд для соответвия программе для работы далее последует сам код для копирования

begin transaction;

define table users schemafull;

define field nick on table users type string;

define field password on table users type string;

define field email on table users type string;

define field description on table users type string;

define index y on table users fields email unique;

define index x on table users fields nick unique;

define table items schemafull;

define field value on table items type int;

define field author on table items;

define field name on table items type string;

define index anames on table items fields name unique;

define field img on table items;

define field description on table items type string;

define field tags on table items type array;

define field tags.\* on table items type string;

define table votes schemafull;

define field item on table votes;

define field user on table votes;

define field value on table votes type int;

define field comment on table votes type string;

define index voted\_already on table votes fields item, user unique;

define table customers schemafull;

define field customer on table customers;

define field adress on table customers type string;

define field executor on table customers;

define field reg\_time on table customers;

define field item on table customers type array;

define field item.\* on table customers type string;

define table tokens schemafull;

define field time on table tokens;

define field user on table tokens;

define field endtime on table tokens;

COMMIT TRANSACTION;

2.2.4 Последовательно работы MVC системы

Model, View, Controller (MVC) — шаблон программирования, разделяющий архитектуру приложения на три модуля: модель (Model), представление (View), контроллер (Controller). Он позволяет изменять каждый компонент независимо друг от друга для простой разработки и поддержки веб-приложений.

Последовательность работы приложения с точки зрения MVC (Рисунок 2.6) показана при помощи диаграммы последовательности UML Sequence Diagram.

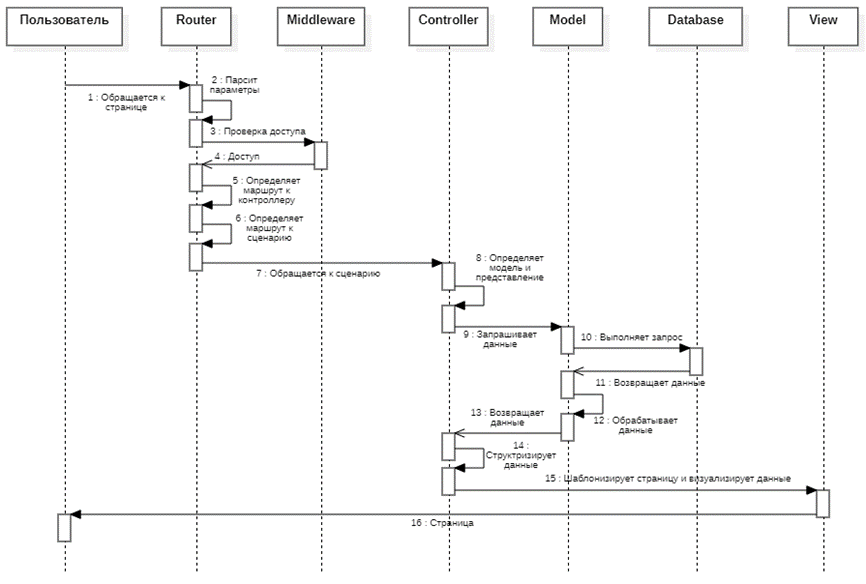


Рис. 2.6. Последовательность работы MVC системы

2.3 Разработка веб-приложения

2.3.1 Организация подключения к СУБД

Для организации подключения к СУБД в MVC используется файл «.ini» в котором хранятся настройки для соединения с базой данных.

Так же php сессия подключаеться к бд через синхронную библиотеку по статическому RPC на порт 8080 выполняясь уже через модели контроллеров

На рисунке 2.7 представлены настройки для подключения к СУБД:

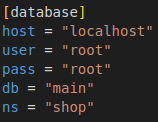
* HOST используется для определения узла сети
* DB используется для определения базы данных
* NS используеться для определения пространства имен
* USERNAME используется для определения пользователя базы данных
* PASS используется для подключения к пользователю базы данных

Рис. 2.7. Настройки для подключения к СУБД

2.3.3 Разработка моделей веб-приложения

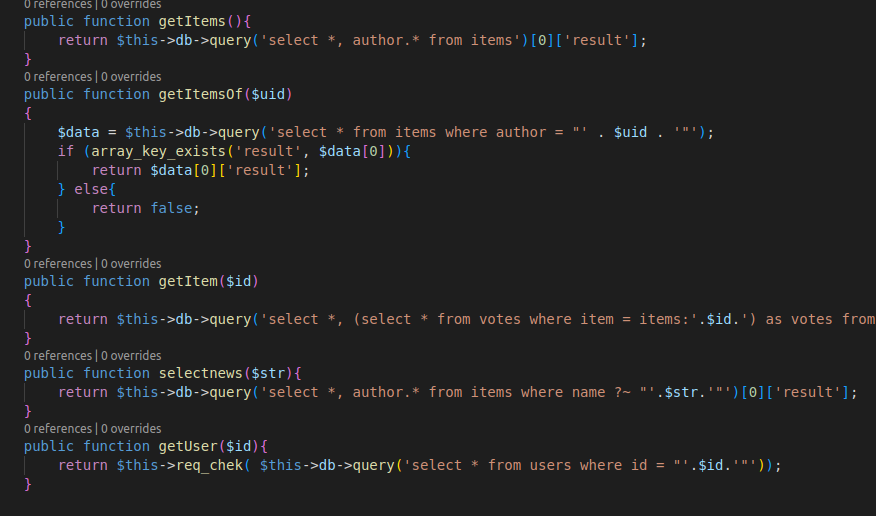
Модель – компонент MVC системы, который отвечает за данные. С помощью модели реализуется статические CRUD функции для работы с данными в базе данных. На рисунке 2.9 представлена модель сущности «main».

Рис. 2.9 Модель сущности «main» базы данных

2.3.4 Разработка контроллеров веб-приложения

Контроллер – это компонент MVC системы, который связывает модель и представление, определяет как сайт реагирует на действия пользователя. Другими словами, это «мозг» MVC-приложения.

**В чистом экземпляре контроллера содержиться**

наследуемый от абстрактного класса **модель** из ядра для каждого контроллера отдельный из этого абстрактного класса.

**view** для вызванного метода контроллера по примеру index должен иметь папу контроллера в этом случае main и файл index.php внутри.

Для вызова представления используеться метод view- > render() в код встроенна документация для дальнейшего рассмотрения.

В системе содержиться 3 контроллера

– main управляет показом основнего контента сайта. показом предметов, пользавателей, выводом результатов поиска или оценок,

так же на него идут все роуты с тегами как ?q=, ?i=, ?p=. по соответвующим методам в нем

q (SearchAction)– поиск пользавателя в строчке

i (ItemAction)– просмотр предмета по id в базе данных

p (Profile\_lookAction) – просмотр пользавателя по его id в базе данных

– account управляет всем что связанно с пользавателем его личный кабинет

– datareq управляет абсолютно всеми запросами клиентов выступая в качестве data API для фронтэнда сайта

2.3.5 Разработка представлений веб-приложения

Одним из требований при разработке веб-приложения была определена разработка простого, понятного интерфейса, позволяющего одинаково и комфортно работать с платформой на различных устройствах.

Верстка интерфейса была сделана с использованием HTML5, CSS3 JS

При верстке были использованы средства медиавыражений CSS (@media), которые позволили регулировать различные аспекты стилей при изменении размера клиентского окна. Таким образом, для пользователя, переход от полноразмерной версии платформы к мобильной происходит незаметно.

Для тестирования графического интерфейса, все элементы интерфейса заполнены данными так, как это должно выглядеть в итоговом варианте приложения.