Для разработки сайта, реализующего следующие возможности для пользователя: регистрация, авторизация, возможность задать вопрос и просмотр всех заданных вопросов; выбран паттерн MVC (Model-view-controller – «модель-вид-контроллер»).

Паттерн MVC включает ряд компонентов:

* Модели определяют структуру и логику используемых данных;
* Представления (views) определяют визуальную часть (каким образом будут отображаться);
* Контроллеры обрабатывают входящие http-запросы, используя для обработки модели и представления, и отправляют в ответ клиенту некоторый результат обработки, нередко в виде html-кода.

**Контроллеры** в паттерне MVC позволяют связать представления и модели и выполняют некоторую логику по обработке запроса. В проекте необходимо определить роутер, который будет сопоставляться с адресами (определить обработку перехода по различным страницам сайта). Отдельно необходимо определить директории для хранения роутеров и контроллеров – routes и controllers соответственно (на данный момент это не реализовано в проекте).

В приложении используется фреймворк Express. Для использования его пакетов необходимо подключить фреймворк в проект:

let express = require('express');

Рассмотрим представленный ниже код:

app.get("/index.html", function (request, response) {

response.sendFile(\_\_dirname + "/public/index.html");

});

app.get("", function (request, response) {

response.sendFile(\_\_dirname + "/public/index.html");

});

app.get("/question.html", function (request, response) {

response.sendFile(\_\_dirname + "/public/question.html");

});

app.get("/question\_read.html", function (request, response) {

get\_question();

response.render('\question\_read', {dataRes: dataRes});

});

app.get("/auth.html", function (request, response) {

response.sendFile(\_\_dirname + "/public/auth.html");

});

app.post("", urlencodedParser, function (request, response) {

if(!request.body) return response.sendStatus(400);

let login = request.body.login;

let email = request.body.email;

let password = request.body.password;

let img = request.file;

add\_user(login, email, password, img.filename, request, response);

});

В данном примере рассмотрено применение обработчиков маршрутов. Маршрутизация определяет, как приложение отвечает на клиентский запрос к конкретному адресу (URI).

Приведенный ниже код обрабатывает переход пользователя на страницу question\_read.html. Когда пользователь переходит на эту страницу, вызывается функция get\_question(), после ее выполнения осуществляется переход на страницу question\_read и передается массив с информацией, считанной из базы данных.

app.get("/question\_read.html", function (request, response) {

get\_question();

response.render('\question\_read', {dataRes: dataRes});

});

**Представления** в рамках проекта созданы для отображения сообщения об успешной авторизации пользователей и чтения вопросов. В качестве движка представлений использован движок EJS. Для хранения представлений создана директория views. Файл success\_auth.ejs содержит код, который позволяет отобразить сообщение об успешной авторизации пользователя и вывести его никнейм. Ниже представлен код файла success\_auth.ejs:

<!DOCTYPE html>

<html >

<head>

<meta charset="UTF-8">

<title>Регистрация</title>

<link rel="stylesheet" href="/public/style.css">

</head>

<body>

<div class="menu">

<button><a href="index.html">Регистрация</a></button>

<button><a href="auth.html">Вход</a></button>

<button><a href="question.html">Задать вопрос</a></button>

<button><a href="question\_read.html">Вопросы</a></button>

</div>

<label>Добро пожаловать,</label>

<label><%= nick %></label>

</body>

</html>

Таким образом, движок ejs позволяет динамически подставить необходимые данные в разметку html-файла.

Файл question\_read.ejs позволяет отобразить заданные вопросы, хранящиеся в базе данных. Ниже представлен код:

<!DOCTYPE html>

<html dir="ltr">

<head>

<meta charset="UTF-8">

<title>Вопросы</title>

<link rel="stylesheet" href="/public/style.css">

</head>

<body>

<div class="menu">

<button><a href="index.html">Регистрация</a></button>

<button><a href="auth.html">Вход</a></button>

<button><a href="question.html">Задать вопрос</a></button>

<button><a href="question\_read.html">Вопросы</a></button>

</div>

<div class="db">

<label>Вопросы</label>

<table border=1 width="80%">

<col style="width:25%">

<col style="width:25%">

<col style="width:30%">

<thead>

</thead>

<tr> <th>Email</th> <th>Вопрос</th> <th>Фото</th>

<% for(var i=0;i<dataRes.length;i++) {%>

<tr><th><%= dataRes[i].email %></th> <th><%= dataRes[i].question %></th><th><img src="/upload/<%= dataRes[i].img %>" width="300px" height="300px"></th></tr>

<% } %>

</table>

</div>

</body>

</html>

При переходе на страницу вопросов выполняется функция считывания информации из таблицы Questions в базе данных. Считанные данные парсятся и записываются в массив dataRes, который передается при вызове функции, выполняющей переход на страницу question\_read. В цикле последовательно выводятся все найденные записи прямо в разметку html.

Таким образом, на данный момент в проекте лишь частично применен паттерн MVC.