**3. Проектно-конструкторская часть**

# 3.1 Разработка структуры программной системы

Приложение представляет собой клиент-серверное приложение, которое является трехуровневым. Клиент-серверное приложение – это вычислительная или сетевая архитектура, в которой задания или сетевая нагрузка распределены между поставщиками услуг, называемыми серверами, и заказчиками услуг, называемыми клиентами. Фактически клиент и сервер — это программное обеспечение.

Трехуровневое клиент-серверное (рисунок 5) приложение содержит в себе:

* Представление данных (клиент). В данной курсовой работе клиентом будет являться веб-браузер, который будет передавать веб-серверу запросы на получение HTML-страниц, изображений, файлов и других данных, обозначенных URL-адресами.
* Сервер приложений располагается на втором уровне, на нём сосредоточена большая часть бизнес-логики. Вне его остаются только фрагменты, экспортируемые на клиента, а также элементы логики, погруженные в базу данных. Реализация данного компонента обеспечивается связующим программным обеспечением.
* Сервер базы данных, выполняет обслуживание и управление базой данных и СУБД, а также отвечает за целостность и сохранность данных и обеспечивает операции ввода-вывода при доступе клиента к информации.

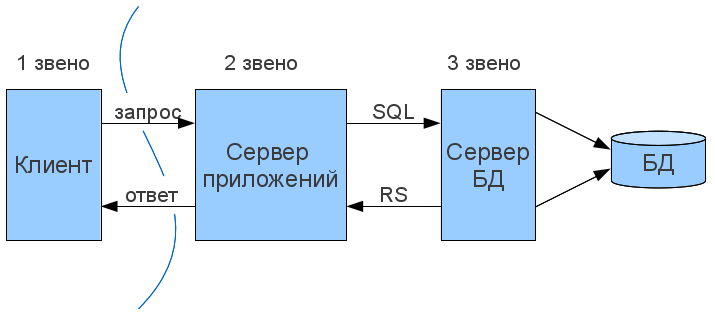


Рисунок 5 – Компоненты системы

Последовательность разработки системы:

1) Проектирование базы данных.

2) Создание логики серверной части.

3) Создание интерфейса.

4) Заполнение базы данных.

5) Тестирование.

# 3.2 Логическая схема данных

Модель «сущность-связь» представляет собой графическую нотацию, которая основывается на блоках и соединительных линиях. С их помощью можно описать объекты модели и отношения между ними. ER имеет несколько основных понятий, таких как сущности, атрибуты и связи. На рисунке 6 представлена логическая схема базы данных разрабатываемой системы.

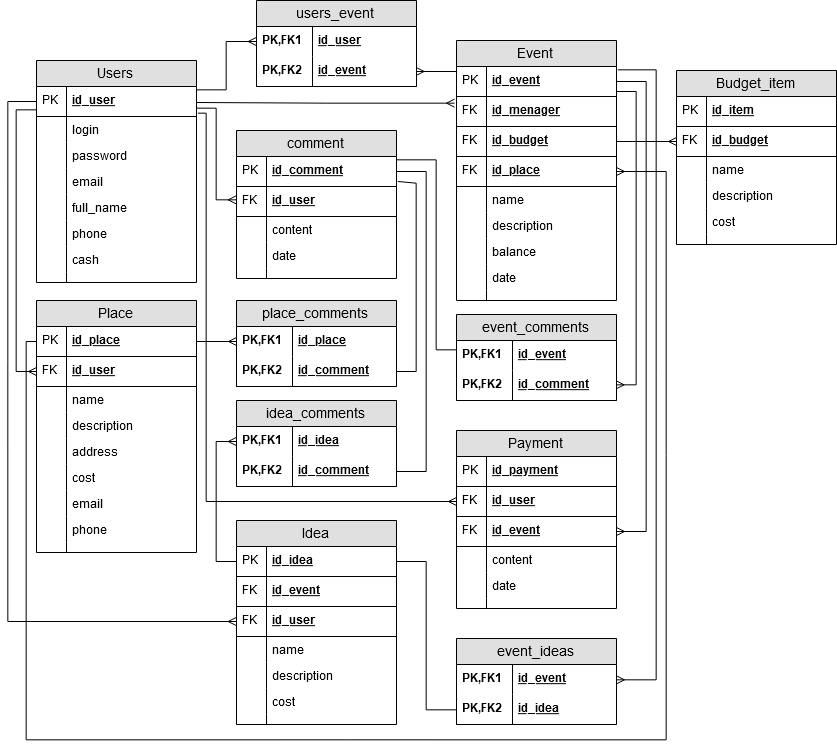


Рисунок 6 – Логическая схема базы данных

В результате анализа предметной области на данном этапе разработки системы были выделены следующие сущности:

Сущностью для хранения данных о участниках, организаторах и арендодателях является Пользователь. Он содержит в себе такие атрибуты как фамилия, имя и отчество, контактный номер и денежный счет. Будучи участником, пользователь может принять участие в мероприятии, оставить комментарий к мероприятиям, идеям, площадкам. Также совершить взнос для участия в мероприятии и создать идею. Поля ФИО, e-mail, контактный телефон необходимо будет заполнить для возможности стать организатором или арендодателем. Как организатор может создать мероприятие, а как арендодатель, может составить информации о сдаваемой в аренду площадке.

Следующей сущностью разрабатываемой системы является Мероприятие. Содержит в себе атрибуты названия, описания, даты проведения и суммы текущих взносов сформированную на основе статей бюджета. Может иметь много участников, только одного организатора и одну площадку. Так как участники принимают участие во многих мероприятиях, а мероприятие содержит много участников, созданная промежуточная таблица пользоватеои-события.

Сущность статья бюджета имеет такие атрибуты как название, описание и стоимость, которую может изменить организатор при получении новой информации. Одна статься бюджета может принадлежать только одному мероприятию.

Сущностью для хранения информации о сдаваемой в аренду площадке является Место, имеет атрибуты: название, описание, адрес, стоимость и контактные данные. Может иметь только одного владельца, а также предоставляться для многих мероприятий.

Для хранения идей пользователей используется сущность Идея, которая имеет атрибуты: название, описание и стоимость. После окончательного принятия идеи организатором информация о ней переходит в сущность Статья бюджета. Идея может иметь отношение только к одному мероприятию и быть инициализирована только одним пользователем.

Сущность Комментарии хранит в себе информацию о комментариях пользователей, такую как содержание и дату комментария. Комментарии могут быть оставлены к событиям, идеям и площадкам. Для связи с вышеперечисленными сущностями используются промежуточные таблицы. Комментарий может относиться только к одной из сущностей.

Для хранения информации о взносах пользователей используется сущность Платеж с атрибутами: описание, количество. Платеж могут совершать все пользователи и только на счет одного мероприятия.

На листинге 1 изображена инициализация подключения к базе данных в настройках сервера Node.js, посредством модуля pg.js.

Листинг 1 – подключение базы данных и проверка посредством запроса текущего времени.

const { Pool} = require('pg');

const connectionString = 'postgresql://postgres:12344321@localhost:5432/evmake\_test'

const pool = new Pool({

connectionString : connectionString,

});

sqlStatus = pool.query('SELECT NOW()', (err, res) => {

if(err){

console.log(err);

}

else{

console.log(res.rows[0]);

}

});

Приложение Node.js запускает сервер и слушает соединения на порте 4000. С помощью методов веб-фреймворка express осуществляются маршрутизация, которая определяет как приложение отвечает на клиентский запрос к конкретному адресу (конечной точке), которым является URI (или путь), и определенному методу запроса HTTP (GET, POST и т.д.). На листинге 2-5 показаны функции обработки запросов GET и POST, которые позволяют читать и записывать информацию о мероприятии из базы данных.

Листинг 2 –данных о событии в формате JSON отправляемые по пути /event/2 посредством POST запроса.

{

"id\_menager" : "1",

"id\_place" : "1",

"name" : "Тестовое событие 2",

"description" : "Описание...",

"date" : "2019-12-26"

}

Листинг 3 – функция обрабатывающая POST запрос для путей event/:id.

app.post('/event/:id', function(req, res){

let event = {

id\_menager : req.body.id\_menager,

id\_place : req.body.id\_place,

name : req.body.name,

description : req.body.description,

date : req.body.date,

}

pool.query(`INSERT INTO events (id\_menager, id\_place, name, description, date) VALUES (${event.id\_menager},

${event.id\_place}, '${event.name}', '${event.description}', '${event.date}');`, function(err){

if(err){

res.send(err);

}

else{

res.sendStatus(200);

}

});

});

Листинг 4 – функция обрабатывающая GET запрос информации о мероприятии.

app.get("/event/:id", function(req, res) {

pool.query(`select \* from events where id\_event = ${req.params.id};`, function(err, result){

if(err){

res.send(err);

}

else {

res.send(result.rows[0]);

}

})

});

Листинг 5 – результат выполнения запроса /event/2.

