ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМ. Р.Е. АЛЕКСЕЕВА»

Институт радиоэлектроники и информационных технологий
Кафедра “Прикладная математика”

Отчет по лабораторной работе курса “Базы данных”

Лабораторная работа №4.1
«Взаимодействие с SQL через HTTP с помощью NodeJS»

Выполнил студент группы 18-ПМ:
Винокуров М.С
Проверил:
Моисеев А.Е.

НИЖНИЙ НОВГОРОД

2020 г.

Оглавление

Введение

NodeJS

Node или **Node.js** — программная платформа, основанная на движке V8 (транслирующем JavaScript в машинный код), превращающая JavaScript из узкоспециализированного языка в язык общего назначения. Node.js добавляет возможность JavaScript взаимодействовать с устройствами ввода-вывода через свой API (написанный на C++), подключать другие внешние библиотеки, написанные на разных языках, обеспечивая вызовы к ним из JavaScript-кода. Node.js применяется преимущественно на сервере, выполняя роль веб-сервера, но есть возможность разрабатывать на Node.js и десктопные оконные приложения (при помощи NW.js, AppJS или Electron для Linux, Windows и macOS) и даже программировать микроконтроллеры (например, tessel и espruino). В основе Node.js лежит событийно-ориентированное и асинхронное (или реактивное) программирование с неблокирующим вводом/выводом.

В состав Node.js входит собственный установщик пакетов npm. Установка производится при помощи команды:

npm install <packagename>

Все доступные для установки пакеты и их краткое описание:

npm search

Этой же командой можно производить выборочный поиск пакетов.

SQLite

SQLite— компактная встраиваемая СУБД. Исходный код библиотеки передан в общественное достояние. В 2005 году проект получил награду Google-O'Reilly Open Source Awards.

Слово «встраиваемый» (embedded) означает, что SQLite не использует парадигму клиент-сервер, то есть движок SQLite не является отдельно работающим процессом, с которым взаимодействует программа, а представляет собой библиотеку, с которой программа компонуется, и движок становится составной частью программы. Таким образом, в качестве протокола обмена используются вызовы функций (API) библиотеки SQLite. Такой подход уменьшает накладные расходы, время отклика и упрощает программу. SQLite хранит всю базу данных (включая определения, таблицы, индексы и данные) в единственном стандартном файле на том компьютере, на котором исполняется программа. Простота реализации достигается за счёт того, что перед началом исполнения транзакции записи весь файл, хранящий базу данных, блокируется; ACID-функции достигаются в том числе за счёт создания файла журнала.

Несколько процессов или потоков могут одновременно без каких-либо проблем читать данные из одной базы. Запись в базу можно осуществить только в том случае, если никаких других запросов в данный момент не обслуживается; в противном случае попытка записи оканчивается неудачей, и в программу возвращается код ошибки. Другим вариантом развития событий является автоматическое повторение попыток записи в течение заданного интервала времени. Сама библиотека SQLite написана на C; существует большое количество привязок к другим языкам программирования, в том числе Apple Swift, Delphi, C++, Java, C#, VB.NET, Python, Perl, Node.js, PHP, PureBasic, Tcl (средства для работы с Tcl включены в комплект поставки SQLite), Ruby, Haskell, Scheme, Smalltalk, Lua и Parser, а также ко многим другим. Полный список существующих средств размещён на странице проекта.

Для NodeJS драйвер SQLite называется *sqlite3* и может быть установлен с помощью *npm.*

Задание

- Создать встраиваемую базу данных из 20-ти объектов
- Вывести на веб-страницу содержимое базы данных (формирование на стороне сервера)

Выполнение работы

Был создан проект Node.JS.
В качестве встраиваемой базы данных была выбрана SQLite.
В качестве объектов хранящихся в базе данных выступает список игр.

В начале программы подключаются необходимые модули:

Для создания веб-сервера

```
var http = require("http");
```

Для работы с базой данных

```
var sqlite3 = require("sqlite3").verbose();
```

Соединение с базой данных

```
var db = new sqlite3.Database("./GameDB.db");
```

Делаем обращение к базе данных, для проверки таблицы

```
db.all("SELECT name FROM sqlite_master WHERE type='table' AND name='GameDB_4';",
function (err, rows)
```

В случае её отсутствия выполняется создание и заполнение таблицы

```
db.run("CREATE TABLE GameDB_4 (game TEXT NOT NULL, price integer NOT NULL, releas
e_date TEXT NOT NULL, platform TEXT NOT NULL);", function (err) {
            //Заполнение
           var statement = db.prepare("INSERT INTO GameDB_4 VALUES(?, ?, ?, ?);")
           statement.run("Fallout", 35, "1994", "Bethesda Launcher");
```

(частичный код программы)

Выполняется создание и логирование в консоль

```
http.createServer(server_callback).listen(3000);
console.log("Listen at http://localhost:3000/");
```

В функции обработки запроса к серверу

```
var server_callback = function (request, response)
```

формируется html документ, в теле оного данные из базы данных

```
var server_callback = function (request, response) {
    response.writeHead(200, { "Content-Type": "text/html; charset=utf-8" });
    response.write("<!DOCTYPE html>\n" +
        "<html>\n" +
        " <head>\n" +
        " <meta charset='utf-8'>\n" +
        " </head>\n" +
        " <body>\n"
    );
    db.all("SELECT * FROM GameDB_4", function(err, rows){
        if (err) {
            response.write("<div style='font-
size: 30px; color:red'>" + err + "</div>\n");
        } else {
            response.write("<table cellspacing=\"2\" border=\"1\" cellpadding=\"5
\">\n");
            for (var i = 0; i < rows.length; i++) {
                response.write("<tr><td>" + rows[i].game + "</td><td align=\"cent
er\">" + rows[i].price + "$</td><td align=\"center\">" + rows[i].release_date + "
</td><td align=\"center\">" + rows[i].platform +"</td></tr>\n");
            }
            response.write("</table>");
        }
        response.end(
            " </body>\n" +
            "</html>\n"
        );
    });
```

Возможно формирование любого SQL-запроса к базе данных, для примера показаны запрос без условий, отображающий всю базу данных:

Fallout	35$	1994	Bethesda Launcher
Fallout 2	45$	1996	Bethesda Launcher
Fallout Nevada	0$	2003	non
Fallout New Vegas	40$	2009	Steam
Halo Combat Evolved	60$	2003	Xbox Live
Halo 2	60$	2006	Xbox Live
Dota 2	0$	2011	Steam
League of legends	0$	2009	RIOT Launcher
Counter Strike - Condition Zero	25$	2005	Steam
Call of Duty 3	60$	2006	Xbox Live
Horizon Zero Dawn	60$	2017	PS Network
The last of Us	60$	2013	PS Network
Fallout 76	50$	2018	Bethesda Launcher
Halo 5	60$	2015	Xbox Live
Stubbs the Zombie	45$	2004	Xbox Live
Battlefield 1942	55$	2003	Retail
Team Fortress 2	5$	2008	Steam
Half-Life	45$	1999	Retail
Half-Life 2	60$	2004	Steam
Half-Life 3	120$	2077	CyberPunkedSteam

И запрос, отображающий игры из магазина «Steam»

Fallout New Vegas	40$	2009	Steam
Dota 2	0$	2011	Steam
Counter Strike - Condition Zero	25$	2005	Steam
Team Fortress 2	5$	2008	Steam
Half-Life 2	60$	2004	Steam

:

Вывод

В ходе выполнения работы был создан простейший веб-сервер, обращающийся к базе данных, формирующий ответ в формате html исходя из данных, хранящихся в базе данных.

Список источников

1. Статья о SQLite в Википедии - https://ru.wikipedia.org/wiki/SQLite
2. Статья о NodeJS в Википедии - https://ru.wikipedia.org/wiki/Node.js
3. Документация NodeJS - https://nodejs.org/en/docs/

Приложение.
Листинг.

GameDB-4.js

```
var http = require("http");
var sqlite3 = require("sqlite3").verbose();
var db = new sqlite3.Database("./GameDB.db");

var server_callback = function (request, response) {
    response.writeHead(200, { "Content-Type": "text/html; charset=utf-8" });
    response.write("<!DOCTYPE html>\n" +
        "<html>\n" +
        " <head>\n" +
        " <meta charset='utf-8'>\n" +
        " </head>\n" +
        " <body>\n"
    );
    //db.all("SELECT * FROM GameDB_4", function(err, rows){
    db.all("SELECT * FROM GameDB_4 WHERE platform='Steam'", function (err, rows)
{

    if (err) {
            response.write("<div style='font-
size: 30px; color:red'>" + err + "</div>\n");
        } else {
            response.write("<table cellspacing=\"2\" border=\"1\" cellpadding=\"5
\">\n");
            for (var i = 0; i < rows.length; i++) {
                response.write("<tr><td>" + rows[i].game + "</td><td align=\"cent
er\">" + rows[i].price + "$</td><td align=\"center\">" + rows[i].release_date + "
</td><td align=\"center\">" + rows[i].platform +"</td></tr>\n");
            }
            response.write("</table>");
        }
        response.end(
            " </body>\n" +
            "</html>\n"
        );
    });

}
db.all("SELECT name FROM sqlite_master WHERE type='table' AND name='GameDB_4';",
function (err, rows) {
    if (err || rows.length == 0) {
        db.run("CREATE TABLE GameDB_4 (game TEXT NOT NULL, price integer NOT NULL
, release_date TEXT NOT NULL, platform TEXT NOT NULL);", function (err) {
            //Заполнение
```

```
            var statement = db.prepare("INSERT INTO GameDB_4 VALUES(?, ?, ?, ?);"
)
            statement.run("Fallout", 35, "1994", "Bethesda Launcher");
            statement.run("Fallout 2", 45, "1996", "Bethesda Launcher");
            statement.run("Fallout Nevada", 0, "2003", "non");
            statement.run("Fallout New Vegas", 40, "2009", "Steam");
            statement.run("Halo Combat Evolved", 60, "2003", "Xbox Live");
            statement.run("Halo 2", 60, "2006", "Xbox Live");
            statement.run("Dota 2", 0, "2011", "Steam");
            statement.run("League of legends", 0, "2009", "RIOT Launcher");
            statement.run("Counter Strike - Condition Zero", 25, "2005", "Steam")
;
            statement.run("Call of Duty 3", 60, "2006", "Xbox Live");
            statement.run("Horizon Zero Dawn", 60, "2017", "PS Network");
            statement.run("The last of Us", 60, "2013", "PS Network");
            statement.run("Fallout 76", 50, "2018", "Bethesda Launcher");
            statement.run("Halo 5", 60, "2015", "Xbox Live");
            statement.run("Stubbs the Zombie", 45, "2004", "Xbox Live");
            statement.run("Battlefield 1942", 55, "2003", "Retail");
            statement.run("Team Fortress 2", 5, "2008", "Steam");
            statement.run("Half-Life", 45, "1999", "Retail");
            statement.run("Half-Life 2", 60, "2004", "Steam");
            statement.run("Half-Life 3", 120, "2077", "CyberPunkedSteam");
            statement.finalize();
        });
        console.log("db GameDB_4 created");
    }
    http.createServer(server_callback).listen(3000);
    console.log("Listen at http://localhost:3000/");
});
```