

Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Кафедра компьютерных систем и программных технологий

Базы данных

Отчет по курсовой работе
Тема работы: "Web-приложение 'Винотека'"

Работу
выполнил:
Графов Д.И.
Группа: 33531/2
Преподаватель:
Мяснов А.В.

Санкт-Петербург
2019

Содержание

1. Техническое задание	3
1.1. Терминология	3
1.2. Постановка задачи	3
1.3. Общение клиента с сервером	3
2. Программа работы	4
3. Реализация	4
3.1. Backend	4
3.2. Frontend	7
4. Вывод	9

1. Техническое задание

1.1. Терминология

Сервер, серверная часть – совокупность скриптов и СУБД, работающих на сервере сервиса. Бэкенд.

Клиентская часть, клиент, приложение – любое стороннее приложение (в данном случае имеется в виду Web-приложение), взаимодействующее с данными сервиса посредством REST API методом отсылки запросов в серверную часть и получения от нее результатов.

CRUD – акроним, обозначающий четыре базовые функции, используемые при работе с базами данных: создание (англ. create), чтение (read), модификация (update), удаление (delete).

1.2. Постановка задачи

Необходимо разработать клиент-серверное приложение для осуществления CRUD операций с таблицей drinks из базы данных (БД) wine_card. В качестве основного языка использовать JavaScript.

Платформой серверной части должна стать Node или Node.js — программная платформа, основанная на движке V8 (транслирующем JavaScript в машинный код).

В качестве сервера стоит использовать сервис, написанный с помощью Express - одного из самых популярных легковесных фреймворков, используемых при создании веб-приложений для node. На серверной части должно быть осуществлено подключение к БД wine_card посредством СУБД PostgreSQL и библиотеки Sequelize.

Клиентская часть должна представлять из себя одностраничное Web-приложение, написанное на Vue.js.

После нажатия на большую кнопку "V" пользователь получает таблицу со всеми напитками, расположенными в базе wine_card. Получив таблицу, пользователь может получить информацию о напитке: его название, рейтинг, объём, крепкость, среднюю цену и тип напитка; изменить существующую запись, удалить её, или добавить новую. Также, после нажатия на элемент таблицы, отображается информация о том, из каких компонентов состоит напиток, а также о том, в каких барах имеется данный напиток.

1.3. Общение клиента с сервером

Общение клиента с сервером осуществлять через http-запросы к /api/ с передачей соответствующих атрибутов. При получении запроса, сервер должен осуществлять соответствующий запрос к СУБД. Далее, если http-запрос является get-запросом, сервер должен отправлять клиенту ответ в формате JSON. Если же это post или delete-запрос, то текстовое сообщение о статусе операции.

2. Программа работы

1. Выбор темы курсовой работы.
2. Написание и согласование технического задания по курсовой работе с подробным описанием реализуемой функциональности.
3. Реализация всей требуемой функциональности.
4. Тестирование корректности работы.
5. Демонстрация результатов преподавателю.
6. Оформление отчета по курсовой работе.

3. Реализация

Со всем исходным кодом приложения можно ознакомиться по адресу:

http://gitlab.icc.spbstu.ru/grafa/wine_card/course.

3.1. Backend

Для написания серверной части была использована библиотека Express.js и Sequelize.js.

Для дальнейших взаимодействий с СУБД посредством библиотеки Sequelize в директории /models были созданы JS-представления таблиц из БД wine_card.

Серверная часть приложения содержит 6 возможных эндпоинтов. (см. листинг 1).

- /api/all_drinks - сервер получает get-запрос, чтобы получить все напитки из таблицы drinks. Обратившись к СУБД, сервер массив объектов в формате json, содержащий всевозможную информацию о напитках.(см. листинг 1, строки 23-29)

Пример возвращаемого объекта:

```
1 {  
2   "drink_id": 1,  
3   "title": "боярский",  
4   "rating": 9.2,  
5   "volume": 100,  
6   "alcohol": 20,  
7   "average_price": 200,  
8   "drink_type": "шоты"  
9 }
```

- /api/components - по get-запросу с атрибутом drink_id сервер обращается к СУБД и возвращает список компонентов данного напитка. Для того, чтобы получить список компонентов, делается запрос к смежной таблице drinks_components, оттуда берётся component_id и по нему ищется вся нужная информация. (см. листинг 1, строки 31-39)
- /api/places - по get-запросу с атрибутом drink_id сервер обращается к СУБД и возвращает список мест, где можно выпить данный напиток. (см. листинг 1, строки 41-49)

- `/api/edit_drink` - по post-запросу с атрибутами `[drink_id, title, rating, volume, alcohol, average_price, drink_type]` сервер обращается к СУБД и изменяет существующую запись в таблице. (см. листинг 1, строки 74-80)
- `/api/delete_drink` - по delete-запросу с атрибутом `drink_id` сервер обращается к СУБД и удаляет существующую запись в таблице. (см. листинг 1, строки 82-100)
- `/api/new_drink` по post-запросу с атрибутами `[drink_id, title, rating, volume, alcohol, average_price, drink_type]` сервер обращается к СУБД и добавляет новую запись в таблицу.

Листинг 1: Исходный код `api.js`

```

1 const express = require('express');
2 const router = express.Router();
3 const path = require('path');
4 const fs = require('fs');
5 const sequelize = require('sequelize');
6 const db = new sequelize(
7   'wine_card', 'postgres', '1234', {
8     host: 'localhost',
9     dialect: 'postgres'
10  }
11 );
12
13 const models = "../models";
14 const root = path.dirname(require.main.filename);
15 fs.readdirSync(`${root}/${models}`).forEach((model) => global[model.substr(0,
16   ↪ model.length - 3)] = db.import(`${root}/${models}/${model}`));
17 console.log("Models_imported_successfully!");
18
19 /* GET users listing. */
20 router.get('/', function (req, res, next) {
21   res.send('respond_with_a_resource');
22 });
23
24 router.get('/all_drinks', async function (req, res, next) {
25   const result = await drinks.findAll(
26     {order: ['drink_id']}
27   );
28   res.set('Access-Control-Allow-Origin', '*');
29   res.json(result);
30 });
31
32 router.get('/components', async function (req, res, next) {
33   const drink_id = req.query.drink_id;
34   const result = await db.query('select c.*
35     from components c
36     join components_drinks cd on cd.
37     ↪ component_id = c.component_id
38     where cd.drink_id = `${drink_id}`');
39   res.set('Access-Control-Allow-Origin', '*');
40   res.json(result[0]);
41 });
42
43 router.get('/places', async function (req, res, next) {
44   const drink_id = req.query.drink_id;
45   const result = await db.query('select p.*
46     from places p

```

```

45                                     join places_drinks pd on pd.place_id
46     ↪ = p.place_id
47     where pd.drink_id = '${drink_id}';
48     res.set('Access-Control-Allow-Origin', '*');
49     res.json(result[0]);
50 });
51 router.post('/edit_drink', async function (req, res) {
52     const title = req.query.title;
53     const volume = req.query.volume;
54     const rating = req.query.rating;
55     const alcohol = req.query.alcohol;
56     const average_price = req.query.average_price;
57     const drink_type = req.query.drink_type;
58
59     drinks.update({
60         title: title,
61         volume: volume,
62         rating: rating,
63         alcohol: alcohol,
64         average_price: average_price,
65         drink_type: drink_type
66     }, {
67         returning: true,
68         where: {drink_id: req.query.drink_id}
69     }).then(res.end("Update_successful!")).catch(function (err) {
70         res.end("Update_failed!")
71     });
72 });
73
74 router.delete('/delete_drink', async function (req, res) {
75     drinks.destroy({
76         where: {drink_id: req.query.drink_id}
77     }).then(res.end("Delete_successful!")).catch(function (err) {
78         res.end("Delete_failed!")
79     })
80 });
81
82 router.post('/new_drink', async function (req, res) {
83     const title = req.query.title;
84     const volume = req.query.volume;
85     const rating = req.query.rating;
86     const alcohol = req.query.alcohol;
87     const average_price = req.query.average_price;
88     const drink_type = req.query.drink_type;
89
90     drinks.create({
91         title: title,
92         volume: volume,
93         rating: rating,
94         alcohol: alcohol,
95         average_price: average_price,
96         drink_type: drink_type
97     }).then(res.end("Insert_successful!")).catch(function (err) {
98         res.end("Insert_failed!")
99     });
100 });
101
102 module.exports = router;

```

3.2. Frontend

Для проектирования пользовательского интерфейса была использована библиотека Vue.js, был написан компонент drinks, который и представляет из себя главную (единственную) страницу web-приложения:

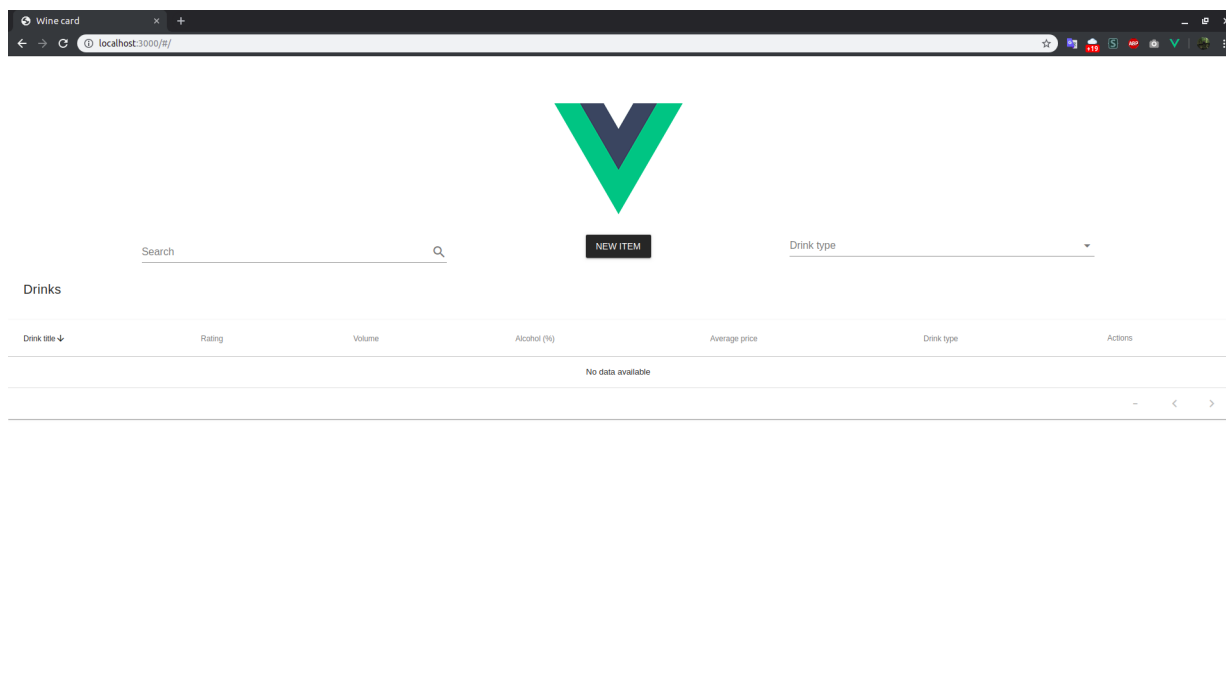


Рисунок 3.1. Начальная страница

При нажатии на кнопку "V" пользователь получает список всевозможных напитков из таблицы drinks:

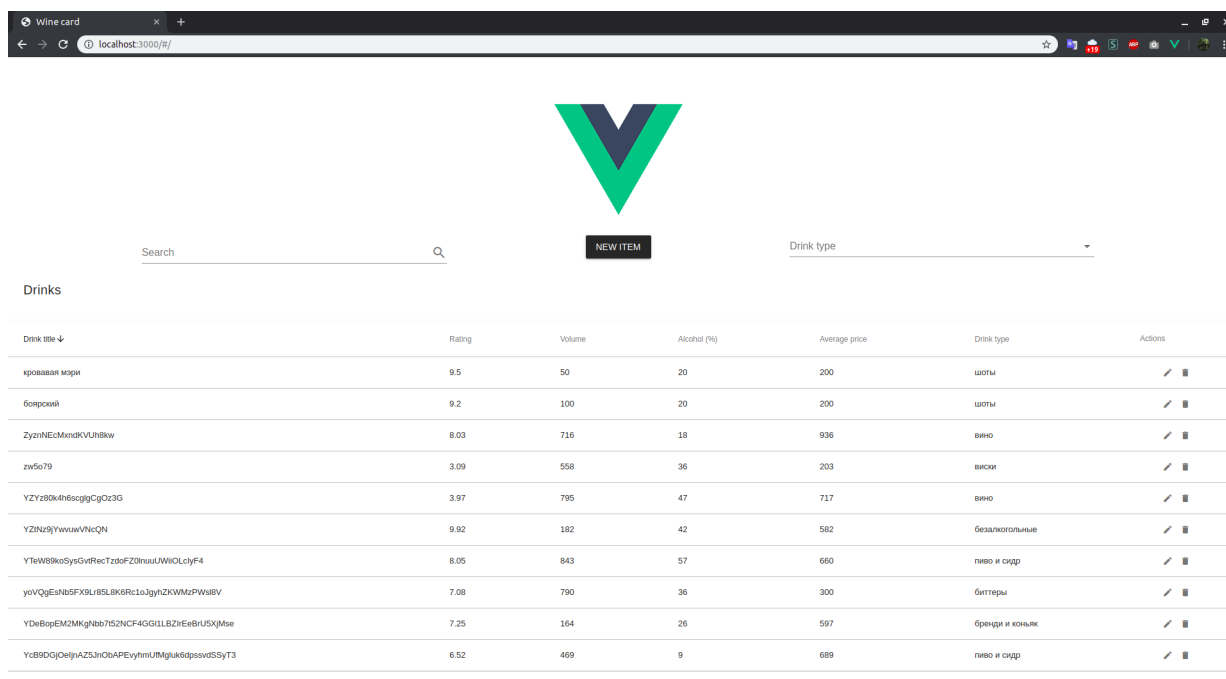


Рисунок 3.2. Начальная страница после нажатия на кнопку "V"

Далее, в строке поиска, пользователь может искать информацию о напитке по его названию.

С помощью кнопки "New item" пользователь может добавить новый напиток в таблицу, указав в модальном окне его параметры:

New Item

Title

Title is required

Alcohol

0

Number must not be null

Volume

0

Number must not be null

Rating

0

Rating must not be empty

Average price

0

Number must not be null

Drink type

CANCEL

SAVE

Рисунок 3.3. Окно добавления нового напитка

При добавлении нового напитка или изменения имеющегося проходит валидация вводимых/изменяемых данных. Осуществляется проверка на то, не являются ли строки пустыми. Для поля title стоит ограничение в 50 символов. Поля, содержащие числовые значения также проверяются. Значения должны быть > 0 . Поле rating должно быть ≤ 10 .

При введении некорректных данных, пользователь не сможет нажать на кнопку "Save" и совершить запрос к серверу.

С помощью нажатия на поле "Drink type" пользователь может из всплывающего списка выбрать тип напитка и отобразить напитки только этого типа:

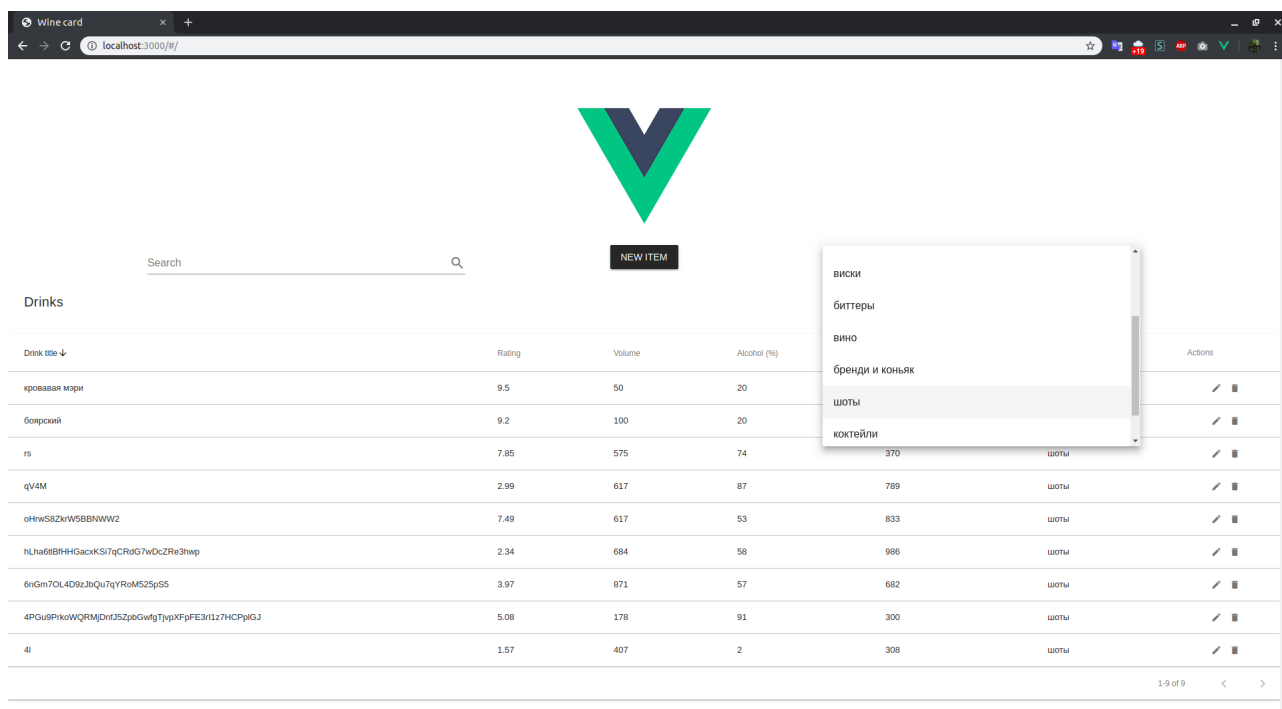


Рисунок 3.4. Сортировка по типу напитка

Если нажать на один из напитков, раскроется информация о том, из каких компонентов он состоит и где можно его выпить:



боярский	9.2	100	20	200	шоты	 
<div> <div>Components: водка гренадин соус табаско</div> <div> Places: <div> <div>Контакт бар по адресу пр. Просвещения, 25</div> <div>Контакт бар по адресу ул. Садовая, 35</div> <div>Контакт бар по адресу пр. Владимирский, 17</div> <div>Контакт бар по адресу пр. Коломяжский, 15, к.2</div> <div>Контакт бар по адресу пл. Стачек, 7, лит. А</div> <div>Контакт бар по адресу ул. Гаккелевская, 34</div> <div>Контакт бар по адресу ул. Марата, 7</div> <div>Контакт бар по адресу Средний пр-т. ВО, 28</div> <div>Контакт бар по адресу пр.Чернышевского 11/57</div> <div>Контакт бар по адресу ул. Бухарестская, 74</div> </div> </div> </div>						

Рисунок 3.5. Дополнительная информация о напитке

Также, пользователь может отредактировать или удалить существующую позицию из таблицы с помощью иконок, расположенных справа.

4. Вывод

В данной курсовой работе мной было реализовано web-приложение, позволяющее через пользовательский интерфейс взаимодействовать с моей БД "wine_card". Мной были использованы изученные мной ранее в семестре хранимые процедуры и триггеры, а также более детально изучена работа с форматом json. Правильность работы моего приложения была протестирована с помощью функционального тестирования.