ФГБОУ ВО «ПЕТРОЗАВОДСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ИНСТИТУТ МАТЕМАТИКИ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Отчет по дисциплине «Методы тестирования ПО»

Выполнила:
студентка группы 22407
Мотина В.С.
Руководитель:
к.ф-м.н., доцент К. А. Кулаков

Петрозаводск 2020

1 Объект тестирования

Средства автоматизации начинают использоваться все более повсеместно, не только на производстве, но и в жизни.

Тестируемое веб-приложение предоставляет возможность учета специй, как дома, так и на предприятии. В доступных пользователям возможностях имеется не только указание базовой информации о специях: наименование, бренд, количество, но и возможность указать её место хранения и добавить теги-примечания к каждой специи. Также в приложении можно использовать поиск по любому заполненному для специи критерию, фильтрацию списков специ, брендов, тегов и мест хранений по всем доступным для заполнения критериям.

Приложение написано при помощи Node.js и Ajax.

Приложение состоит из одной страницы, но имеет следующие вкладки:

1. Мои специи - на вкладке доступен список специй, имеющихся у пользователя, их количество и место хранения. А также возможность добавить, редактировать и удалить любую позицию списка.
2. Специи - на вкладке доступен список специй, добавленных в систему, их описание, бренд и тег. А также возможность добавить, редактировать и удалить любую позицию списка.
3. Хранилища - на вкладке доступен список мест хранения, добавленных пользователем, и их описание. А также возможность добавить, редактировать и удалить любую позицию списка.
4. Бренды на вкладке доступен список брендов, добавленных пользователем, и их описание. А также возможность добавить, редактировать и удалить любую позицию списка.
5. Теги на вкладке доступен список брендов, добавленных пользователем. А также возможность добавить, редактировать и удалить любую позицию списка.

1.1 Требования

T1: Приложение должно запускать в браузере FIrefox, не ниже 78.4.1 версии.
T2: Передача данных должна происходить по rest api.
T3: Веб приложение должно иметь адаптивную верстку.
T4: На главной странице должны присутствовать все вкладки, для перемещения между ними.
T5: При нажатии на кнопку создания элемента списка специй и при редактировании элемента списка, должно открываться диалоговое окно с формой для заполнения.
T6: При создании нового бренда, тега и места хранения поля заполняется в первой строке списка.
T7: При удалении бренда, должны удаляться все связанные с ним специи.

1.2 Функциональнальные требования объекта тестирования

1. Управление брендами.
    a. Добавление бренда.
    b. Редактирование бренда.
    c. Удаление бренда.
    d. Просмотр списка всех добавленных брендов.
2. Управление местами хранения.
    a. Добавление места хранения.
    b. Редактирование места хранения.
    c. Удаление места хранения.
    d. Просмотр списка всех добавленных мест хранения.
3. Управление тегами.
    a. Добавление тега.
    b. Редактирование тега.
    c. Удаление тега.
    d. Просмотр списка всех добавленных тегов.
4. Управление специями.
    a. Добавление специи.
    b. Редактирование специи.
    c. Удаление специи.
    d. Просмотр списка всех добавленных специй.

В аттестационном тестировании принимают участие все вышеописанные функции.

Не реализованные:

5. Поиск специй по:
    a. бренду,
    b. тегу,
    c. месту расположения,
    d. названию специи
6. Фильтрация специй по:
    a. бренду,
    b. тегу,
    c. месту расположения,
    d. названию специи.

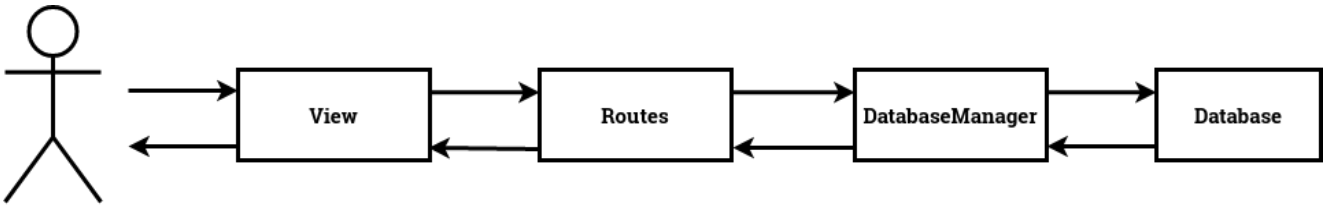
2 Стратегия тестирования

2.1 Описание структуры объекта тестирования и связей внутри объекта тестирования (архитектура)

2.1.1 Структура кода приложения

├─ controllers - каталог файлов маршрутов
├─ models - каталог файлов взаимодействия с базой данных
├─ node_modules - библиотеки для node.js
├─ public - каталог файлов ajax
├─ databaseManager.js
├─ server.js - главный файл
└─ spices.sql - файл создание пустой базы данных

2.1.2 Архитектура приложения



2.1.2 Модули приложения

- View - модуль отображения
- Routers - модуль маршрутизации
- DatabaseManager - интерфейс к базе данных2.2 Основные положения процедуры проведения тестирования

2.2 Основные положения процедуры проведения тестирования

1. Тест считается успешно пройденным, если ожидаемый и фактический результат теста совпадают.
2. Этап тестирования считается завершенным в том случае, когда 50 % тестов текущего этапа успешно пройдены.
3. Переход к следующему этапу тестирования может быть совершен только в том случае, если предыдущий этап тестирования завершен.
4. Процедура проведения тестирования может быть остановлена только в случае ошибки, блокирующей выполнение тестов.

5. Возобновление процедуры проведения тестирования происходит после исправления блокирующей ошибки, которая помешала предыдущему выполнению.

2.3 Инструменты тестирования

Для юзабилити тестов:

- Ручное тестирование

Для блочных и интеграционных тестов:

- Postman - инструмент тестирования API
- cUrl - инструмент тестирования API
- jest - инструмент тестирования js проектов
- superTest - подмена запросов

Для специальных тестов:

- Самописный скрипт генерации данных и ручное тестирование

Для аттестационных тестов:

- Ручное тестирование

2.4 Стратегия блочного тестирования

1. Тестирование проекта начинается от тестирования отдельных изолированных(в контролируемом окружении) компонентов - с блочного тестирования.
2. Данный вид тестирования будет применен к всем функциям модуля DatabaseManager.tag-model.
3. Использование средств автоматизации тестирования - jest.
4. Использование классов-заглушек.

Тестируемый модуль - DatabaseManager.tag-model:

- **deleteTag()**
    - **Описание:** Удаление тега
    - **Входные параметры:** tag_id
    - **Выход:** статус
- **addTag(tag_title)**
    - **Описание:** Добавление тега
    - **Входные параметры:** tag_title
    - **Выход:** статус, tag_id и tag_title добавленного тега
- **editTag(tag_id, tag_title)**
    - **Описание:** Редактирование тега
    - **Входные параметры:** tag_id, tag_title
    - **Выход:** статус, tag_id и tag_title отредактированного тега
- **getAllTags()**
    - **Описание:** Получение списка всех тегов

    - **Входные параметры:** отсутствуют
    - **Выход:** статус, массив объектов, включающих tag_id и tag_title

Тестируемый модуль - Controllers.tag-router

- **post()**
    - **Описание:** Добавление тега
    - **Входные параметры:** tag_title
    - **Выход:** статус, {tag_id, tag_title}
- **put()**
    - **Описание:** Редактирование тега
    - **Входные параметры:** tag_id, tag_title
    - **Выход:** статус, {tag_id, tag_title}
- **get()**
    - **Описание:** Получение списка тегов
    - **Входные параметры:** отсутствуют
    - **Выход:** статус, объект состоящий из объектов, содержащих tag_id и tag_title
- **delete()**
    - **Описание:** Удаление тега
    - **Входные параметры:** tag_id
    - **Выход:** статус

2.5 Стратегия интеграционного тестирования

1. Проверка взаимодействия компонент
2. Данный вид тестов будет применен ко всем функциям Controller.tag-router и DatabaseManager.tag-model.
3. Использование тестовой базы данных, обновляемой перед каждым тестом.
4. Использование средств автоматизации тестирования - jest. И библиотеки для формирования запросов superTest.

   Взаимодействие модулей: Controllers и DatabaseManager для тегов

- Начало теста --> supertest --> controllers.tag-router.post() --> databaseManager.tag-model.addTag() --> TestDatabase --> databaseManager.callback --> Controllers.callback --> Конец теста
- Начало теста --> supertest(отправка запроса на получение) --> controllers.tag-router.post() --> databaseManager.tag-model.getAllTag() --> TestDatabase --> databaseManager.callback --> Controllers.callback --> Конец теста
- Начало теста --> supertest(отправка запроса на редактирование) --> controllers.tag-router.post() --> databaseManager.tag-model.editTag() --> TestDatabase --> databaseManager.callback --> Controllers.callback --> Конец теста
- Начало теста --> supertest(отправка запроса на удаление) --> controllers.tag-router.post() --> databaseManager.tag-model.deleteTag() --> TestDatabase --> databaseManager.callback --> Controllers.callback --> Конец теста

2.6 Стратегия юзабилити тестирования

Юзабилити тестирование - исследование, проводимое с целью определения удобства взаимодействия с предлагаемым сервисом. Для повышения скорости освоения в системе, через создание доступного для понимания и освоения интерфейса.

2.7 Стратегия специального тестирования

Нагрузочное тестирование – это процесс умышленной нагрузки системы, с целью определения показателей производительности, времени отклика, проверки соответствия требованиям, которые были предъявлены к данной системе.

Целью данного тестирования является оценка производительности и работоспособности тестируемого модуля. Производительность, время отклика и соответствие требованиям исследуется при отображении результата на разных нагрузках в широких диапазонах.

Для данного проекта нагрузочное тестирование будет исследоваться по следующему критерию: отображение значительного количества элементов списков, оценка времени работы. Проводится для наиболее загруженного информацией и зависимостями списка “Специи”, следующим образом:

1. Для заполнения базы данных будет написан скрипт на js для генерации N элементов тестовых данных в формате sql, с выгрузкой данных в отдельный файл.
2. Последовательно добавляется недостающее до следующего порога число объектов.
3. Этапы увеличения нагрузки:

Этап	**Число объектов**
Первый	10
Второй	100
Третий	250
Четвертый	1000

2.8 Стратегия аттестационного тестирования

В ходе аттестационного тестирования будет проведено оценочное тестирование работоспособности реализованных в приложении на данный момент функций со стороны пользователя. Оценка соответствия приложения и заявленного функционала.

Человек, проводящий тестирование, следуя заранее написанному алгоритму, воспроизводит действия и сверяет результат с ожидаемым результатом.

2.9 Способы оценивания результатов прохождения тестов

2.9.1 Критерии прохождения тестов

Тест, у которого ожидаемый результат и фактический результат совпадут, считается успешно пройденным.

2.9.2 Критерий приостановления тестирования

Тестирование должно быть приостановлено, если количество не пройденных тестов превысит 20% от общего количества.

Если тест завершился неудачно, необходимо проверить тест на достоверность, после чего, если он все ещё завершается неудачей, предоставить отчет об ошибке.

2.9.3 Критерий возобновления работы

Необходимо начинать тестирование заново после каждого итерации разработки, чтобы убедиться в исправлении опознанных ошибок и отлове появившихся новых.

3 Детальный план тестов

3.1 Корректность возвращаемых данных

Корректно работающий сервер:

1. Доступный
2. Без ошибок в логике работы
3. Возвращает валидные данные в формате JSON, коды ответа сервера должны быть в диапазоне 200-399. Пример:

```
[
  RowDataPacket {
    spice_id: 5,
    spice_title: 'Для сочного мяса с чесноком и луком по-домашнему',
    store_id: 2,
    store_title: '2 контейнер',
    batch_id: 4,
    batch_number: '2'
  },
  RowDataPacket {
    spice_id: 5,
    spice_title: 'Для шаурмы по-домашнему',
    store_id: 1,
    store_title: '1 контейнер',
    batch_id: 2,
    batch_number: '1'
  }
]
```

3.2 План блочного тестирования

Модуль DatabaseManager.tag-model

Тест Б1
Тип: позитивный
Описание: проверка удаления существующего тега из базы данных.
Функция: deleteTag()
Входные данные: tag_id=18.
Ожидаемый результат: переданная функция обработчик будет вызвана со значением 1, будет создано два запроса к базе, созданное соединение будет закрыто.

Тест Б2
Тип: негативный
Описание: проверка удаления несуществующего тега из базы данных.
Функция: deleteTag()
Входные данные: tag_id=20.
Ожидаемый результат: возвращаемый результат: status=400, созданное соединение будет закрыто.

Тест Б3
Тип: позитивный
Описание: проверка добавления не пустого тега.
Функция: addTag()
Входные данные: tag_title=”NewTegTitle”.
Ожидаемый результат: будет создано три запроса к базе, возвращаемый результат: status=200, {tag_id=19, tag_title=”NewTegTitle”}, созданное соединение будет закрыто.

Тест Б4
Тип: негативный
Описание: проверка добавления тега с пустым именем.
Функция: addTag()
Входные данные: tag_title=””.
Ожидаемый результат: возвращаемый результат: status=400, созданное соединение будет закрыто.

Тест Б5
Тип: позитивный
Описание: проверка редактирования существующего тега.
Функция: editTag()
Входные данные: tag_id=18, tag_title=”NewTegTitle”.
Ожидаемый результат: будет создано два запроса к базе, возвращаемый результат: status=200, {tag_id=18, tag_title=”NewTegTitle”}, созданное соединение будет закрыто.

Тест Б6
Тип: негативный
Описание: проверка редактирования не существующего тега.
Функция: editTag()
Входные данные: tag_id=19, tag_title=”NewTegTitle”.
Ожидаемый результат: будет создано два запроса к базе, возвращаемый результат: status=400, созданное соединение будет закрыто.

Тест Б7
Тип: позитивный
Описание: проверка получения списка всех тегов.
Функция: getAllTags()
Входные данные: отсутствуют.
Ожидаемый результат: будет создан один запрос к базе, возвращаемый результат: status=200, [{ tag_id: 1, tag_title: 'miaso' },{ tag_id: 2, tag_title: 'chicken' },{ tag_id: 3, tag_title: 'fish' },{ tag_id: 5, tag_title: 'fffnc' },{ tag_id: 6, tag_title: 'ggg' },{ tag_id: 16, tag_title: 'мясо' },{ tag_id: 17, tag_title: 'макароны' },{ tag_id: 18, tag_title: 'плов' }] , созданное соединение будет закрыто.

Модуль Controllers.tag-router

Тест Б8
Тип: позитивный
Описание: добавление тега
Метод: post()
Входные данные: tag_title=“NewTegTitle”
Ожидаемый результат: status=200, {tag_id=19, tag_title=“NewTegTitle”}

Тест Б9
Тип: негативный
Описание: добавление тега с пустым названием
Метод: post()
Входные данные: tag_title=“”
Ожидаемый результат: status=400

Тест Б10
Тип: позитивный
Описание: получение списка всех тегов
Метод: get()
Входные данные: отсутствуют
Ожидаемый результат: {{ tag_id: 1, tag_title: 'miaso' },{ tag_id: 2, tag_title: 'chicken' },{ tag_id: 3, tag_title: 'fish' },{ tag_id: 5, tag_title: 'fffnc' },{ tag_id: 6, tag_title: 'ggg' },{ tag_id: 16, tag_title: 'мясо' },{ tag_id: 17, tag_title: 'макароны' },{ tag_id: 18, tag_title: 'плов' }}

Тест Б11
Тип: позитивный
Описание: редактирование существующего тега

Метод: put()
Входные данные: tag_id=18, tag_title="NewTegTitle"
Ожидаемый результат: status=200, {tag_id=18, tag_title="NewTegTitle"}

Тест Б12
Тип: негативный
Описание: редактирование не существующего тега
Метод: put()
Входные данные: tag_id=19, tag_title="NewTegTitle"
Ожидаемый результат: status=400

Тест Б13
Тип: позитивный
Описание: удаление существующего тега
Метод: put()
Входные данные: tag_id=18
Ожидаемый результат: status=200

Тест Б14
Тип: негативный
Описание: удаление не существующего тега
Метод: put()
Входные данные: tag_id=19
Ожидаемый результат: status=400

3.3 План интеграционного тестирования

Тест И1
Тип: позитивный
Описание: проверка добавления не пустого тега
Условия выполнения: Начало теста -(1)-> supertest(отправка запроса на добавление) -(2)-> controllers.tag-router.post() -(3)-> databaseManager.tag-model.addTag() -(4)-> TestDatabase -(5)-> databaseManager.callback -(6)-> Controllers.callback -(7)-> Конец теста
Объект тестирования: Взаимодействие между Controllers и DatabaseManager
Данные на этапах:
(2) {tag_title: "newTestTag"}
(3) параметры: title = "newTestTag"
(6) {tag_id: "19", tag_title: "newTestTag"}
(7) {tag_id: "19", tag_title: "newTestTag"}, status=200
Ожидаемый результат: Входные параметры переданы, возвращаемое значение включает входные параметры и новые данные: идентификатор созданного тега

Тест И2
Тип: негативный
Описание: проверка добавления пустого тега

Условия выполнения: Начало теста -(1)-> supertest(отправка запроса на добавление) -(2)-> controllers.tag-router.post() -(3)-> databaseManager.tag-model.addTag() -(4)-> databaseManager.callback -(6)-> Controllers.callback -(7)-> Конец теста

Объект тестирования: Взаимодействие между Controllers и DatabaseManager

Данные на этапах:

(2) {tag_title: “”}

(3) параметры: title = “”

(6) status=400

(7) status=400

Ожидаемый результат: Ошибка добавления

Тест И3

Тип: позитивный

Описание: проверка получения всех тегов

Условия выполнения: Начало теста -(1)-> supertest(отправка запроса на получение) -(2)-> controllers.tag-router.post() -(3)-> databaseManager.tag-model.getAllTag() -(4)-> TestDatabase -(5)-> databaseManager.callback -(6)-> Controllers.callback -(7)-> Конец теста

Объект тестирования: Взаимодействие между Controllers и DatabaseManager

Данные на этапах:

(6)[{ tag_id: 1, tag_title: 'miaso' }, { tag_id: 2, tag_title: 'chicken' }, { tag_id: 3, tag_title: 'fish' }, { tag_id: 5, tag_title: 'fffnc' }, { tag_id: 6, tag_title: 'ggg' }, { tag_id: 16, tag_title: 'мясо' }, { tag_id: 17, tag_title: 'макароны' }, { tag_id: 18, tag_title: 'плов' }]

(7) {{ tag_id: 1, tag_title: 'miaso' }, { tag_id: 2, tag_title: 'chicken' }, { tag_id: 3, tag_title: 'fish' }, { tag_id: 5, tag_title: 'fffnc' }, { tag_id: 6, tag_title: 'ggg' }, { tag_id: 16, tag_title: 'мясо' }, { tag_id: 17, tag_title: 'макароны' }, { tag_id: 18, tag_title: 'плов' }}

Ожидаемый результат: Получение списка тегов: {{ tag_id: 1, tag_title: 'miaso' }, { tag_id: 2, tag_title: 'chicken' }, { tag_id: 3, tag_title: 'fish' }, { tag_id: 5, tag_title: 'fffnc' }, { tag_id: 6, tag_title: 'ggg' }, { tag_id: 16, tag_title: 'мясо' }, { tag_id: 17, tag_title: 'макароны' }, { tag_id: 18, tag_title: 'плов' }}

Тест И4

Тип: позитивный

Описание: проверка редактирования существующего тега на непустое значение

Условия выполнения: Начало теста -(1)-> supertest(отправка запроса на редактирование) -(2)-> controllers.tag-router.post() -(3)-> databaseManager.tag-model.editTag() -(4)-> TestDatabase -(5)-> databaseManager.callback -(6)-> Controllers.callback -(7)-> Конец теста

Объект тестирования: Взаимодействие между Controllers и DatabaseManager

Данные на этапах:

(2) {tag_id: “18”, tag_title: “newTestTag”}

(3) параметры: id = 18, title = “newTestTag”

(6) {tag_id: “18”, tag_title: “newTestTag”}

(7) {tag_id: “18”, tag_title: “newTestTag”}, status=200

Ожидаемый результат: Входные параметры переданы, возвращаемое значение совпадает со входными параметрами

Тест И5

Тип: негативный

Описание: проверка редактирования не существующего тега на не пустое значение

Условия выполнения: Начало теста -(1)-> supertest(отправка запроса на редактирование) -(2)-> controllers.tag-router.post() -(3)-> databaseManager.tag-model.editTag() -(4)-> TestDatabase -(5)-> databaseManager.callback -(6)-> Controllers.callback -(7)-> Конец теста

Объект тестирования: Взаимодействие между Controllers и DatabaseManager

Данные на этапах:

(2) {tag_id: “20”, tag_title: “newTestTag”}

(3) параметры: id = 20, title = “newTestTag”

(6) status=400

(7) status=400

Ожидаемый результат: Ошибка редактирования

Тест И6

Тип: негативный

Описание: проверка редактирования существующего тега на пустое значение

Условия выполнения: Начало теста -(1)-> supertest(отправка запроса на редактирование) -(2)-> controllers.tag-router.post() -(3)-> databaseManager.tag-model.editTag() -(4)-> TestDatabase -(5)-> databaseManager.callback -(6)-> Controllers.callback -(7)-> Конец теста

Объект тестирования: Взаимодействие между Controllers и DatabaseManager

Данные на этапах:

(2) {tag_id: “20”, tag_title: “”}

(3) параметры: id = 20, title = “”

(6) status=400

(7) status=400

Ожидаемый результат: Ошибка редактирования

Тест И7

Тип: позитивный

Описание: проверка удаления существующего тега

Условия выполнения: Начало теста -(1)-> supertest(отправка запроса на удаление) -(2)-> controllers.tag-router.post() -(3)-> databaseManager.tag-model.deleteTag() -(4)-> TestDatabase -(5)-> databaseManager.callback -(6)-> Controllers.callback -(7)-> Конец теста

Объект тестирования: Взаимодействие между Controllers и DatabaseManager

Данные на этапах:

(2) {tag_id: “18”}

(3) параметр: id = 18

(6) {tag_id: “18”}, status=200

(7) {tag_id: “18”}, status=200

Ожидаемый результат: Входные параметры переданы, возвращаемое значение совпадает со входными параметрами, успешное удаление

Тест И8

Тип: негативный

Описание: проверка удаления несуществующего тега

Условия выполнения: Начало теста -(1)-> supertest(отправка запроса на удаление) -(2)-> controllers.tag-router.post() -(3)-> databaseManager.tag-model.deleteTag() -(4)-> TestDatabase -(5)-> databaseManager.callback -(6)-> Controllers.callback -(7)-> Конец теста

Объект тестирования: Взаимодействие между Controllers и DatabaseManager

Данные на этапах:

(2) {tag_id: “20”}

(3) параметр: id = 20

(6) status=400

(7) status=400

Ожидаемый результат: Ошибка удаления

3.4 План юзабилити тестирования

Тест Ю1

Цель: Проверка масштабируемости страницы

Предсостояние: главная страница

Действия: плавное изменение размера окна от полноразмерного до минимально возможного

Ожидаемый результат: плавное изменение размера страницы, без выхода за пределы зоны видимости содержимого. Переход вкладок из горизонтального порядка в вертикальный

Тест Ю2

Цель: Проверка обновления данных

Предсостояние: вкладка “Специи”

Действия: удаление элемента из списка

Ожидаемый результат: после подтверждения удаления, строка удаляется из списка без перезагрузки страницы

Тест Ю3

Цель: Проверка обновления данных

Предсостояние: вкладка “Специи”

Действия: добавление элемента

Ожидаемый результат: после заполнения формы на добавление, в конце списка появляется новая строка.

Тест Ю4

Цель: Проверка перехода между вкладками

Предсостояние: вкладка “Специи”

Действия: переход на вкладку “Теги”

Ожидаемый результат: переход на вкладку “Теги” без перезагрузки страницы.

Тест Ю4

Цель: Проверка редактирования специи
Предсостояние: вкладка “Специи”
Действия: нажать на строку в списке
Ожидаемый результат: открытие диалогового окна для редактирования.

Тест Ю5
Цель: Проверка редактирования тега
Предсостояние: вкладка “Теги”
Действия: нажать на строку в списке
Ожидаемый результат: в выбранной строке, можно изменить поле “название”.

Тест Ю6
Цель: Проверка редактирования места хранения
Предсостояние: вкладка “Хранилища”
Действия: нажать на строку в списке
Ожидаемый результат: в выбранной строке, можно изменить поле “название” и “описание”.

Тест Ю7
Цель: Проверка редактирования бренда
Предсостояние: вкладка “Бренды”
Действия: нажать на строку в списке
Ожидаемый результат: в выбранной строке, можно изменить поле “название” и “описание”.

Тест Ю8
Цель: Проверка удаление любой строки
Предсостояние: любая, например вкладка “Специи”
Действия: нажать на символ “мусорной корзины” в строке из списка
Ожидаемый результат: появление диалогового окна с подтверждением удаления.

Тест Ю9
Цель: Проверка добавления специи
Предсостояние: вкладка “Специи”
Действия: нажать на кнопку “Новая специя”
Ожидаемый результат: появление диалогового окна с формой на добавление специи.

Тест Ю10
Цель: Проверка добавления специи
Предсостояние: вкладка “Специи”, диалоговое окно добавления специи
Действия: заполнить форму, нажать на кнопку “Сохранить”
Ожидаемый результат: появление новой строки с заданными параметрами в конце списка.

3.2 План специального тестирования

Тест С1
Цель: Проверка отображения списка с малым количеством записей
Тип: общий
Входные данные: список имеющий 10 записей
Ожидаемый результат: отображение 10 записей в списке

Тест С2
Цель: Проверка отображения списка с небольшим количеством записей
Тип: общий
Входные данные: список имеющий 100 записей
Ожидаемый результат: отображение 100 записей в списке

Тест С3
Цель: Проверка отображения списка со средним количеством записей
Тип: общий
Входные данные: список имеющий 250 записей
Ожидаемый результат: отображение 250 записей в списке

Тест С4
Цель: Проверка отображения списка с малым количеством записей
Тип: общий
Входные данные: список имеющий 1000 записей
Ожидаемый результат: отображение 1000 записей в списке

3.4 План аттестационного тестирования

3.4.1 Тестирование брендов

Тест А1
Цель: Проверка добавления бренда
Тип: позитивный
Алгоритм проведения:

- Открыть веб-приложение
- Открыть вкладку “Бренды”
- Добавить бренд, заполнив поле “название” используя не более 255 символов.

Ожидаемый результат: добавление нового бренда в список брендов
Тест А2
Цель: Проверка добавления бренда
Тип: негативный
Алгоритм проведения:

- Открыть веб-приложение
- Открыть вкладку “Бренды”

- Добавить бренд, заполнив поле “название” используя более 255 символов.

Ожидаемый результат: вывод ошибки добавления.

Тест А3
Цель: Редактирование бренда
Тип: позитивный
Алгоритм проведения:

- Открыть веб-приложение
- Открыть вкладку “Бренды”
- Изменить “название” бренда используя не более 255 символов.

Ожидаемый результат: изменение названия бренда в список брендов.

Тест А4
Цель: Редактирование бренда
Тип: негативный
Алгоритм проведения:

- Открыть веб-приложение
- Открыть вкладку “Бренды”
- Изменить “название” бренда используя не более 255 символов.

Ожидаемый результат: вывод ошибки редактирования.

Тест А5
Цель: удаление бренда
Тип: позитивный
Алгоритм проведения:

- Открыть веб-приложение
- Открыть вкладку “Бренды”
- выбрать бренд из списка
- подтвердить удаление

Ожидаемый результат: Удаление бренда из списка брендов, удаление всех специй, у которых был указан удаленный бренд.

3.4.2 Тестирование тегов

Тест А6
Цель: Проверка добавления тега
Тип: позитивный
Алгоритм проведения:

- Открыть веб-приложение
- Открыть вкладку “Теги”
- Добавить тег, заполнив поле “название” используя не более 255 символов.

Ожидаемый результат: добавление нового тега в список тегов.

Тест А7
Цель: Проверка добавления тега
Тип: негативный

Алгоритм проведения:

- Открыть веб-приложение
- Открыть вкладку “Теги”
- Добавить тег, заполнив поле “название” используя более 255 символов.

Ожидаемый результат: ошибка добавления.

Тест А8
Цель: Проверка редактирования тега
Тип: позитивный
Алгоритм проведения:

- Открыть веб-приложение
- Открыть вкладку “Теги”
- Редактировать тег, заполнив поле “название” используя не более 255 символов.

Ожидаемый результат: изменение названия тега в списке тегов.

Тест А9
Цель: Проверка редактирования тега
Тип: негативный
Алгоритм проведения:

- Открыть веб-приложение
- Открыть вкладку “Теги”
- Редактировать тег, заполнив поле “название” используя более 255 символов.

Ожидаемый результат: ошибка редактирования.

Тест А10
Цель: Проверка удаление тега
Тип: позитивный
Алгоритм проведения:

- Открыть веб-приложение
- Открыть вкладку “Теги”
- Выбрать тег для удаления из списка
- подтвердить удаление

Ожидаемый результат: успешное удаление тега из списка тегов.

3.4.3 Тестирование хранилища

Тест А11
Цель: Проверка добавления места хранения
Тип: позитивный
Алгоритм проведения:

- Открыть веб-приложение
- Открыть вкладку “Хранилища”
- Добавить место хранения, заполнив поле “название” используя не более 255 символов.

Ожидаемый результат: добавление нового тега в список тегов.

Тест А12
Цель: Проверка добавления места хранения
Тип: негативный
Алгоритм проведения:

- Открыть веб-приложение
- Открыть вкладку “Хранилища”
- Добавить место хранения, заполнив поле “название” используя более 255 символов.

Ожидаемый результат: ошибка добавления.

Тест А13
Цель: Проверка редактирования места хранения
Тип: позитивный
Алгоритм проведения:

- Открыть веб-приложение
- Открыть вкладку “Хранилища”
- Редактировать место хранения, изменив поле “название” используя не более 255 символов.

Ожидаемый результат: изменение названия места хранения в списке хранилищ.

Тест А14
Цель: Проверка редактирования места хранения
Тип: негативный
Алгоритм проведения:

- Открыть веб-приложение
- Открыть вкладку “Хранилища”
- Редактировать место хранения, изменив поле “название” используя более 255 символов.

Ожидаемый результат: ошибка редактирования.

Тест А15
Цель: Проверка удаления места хранения
Тип: позитивный
Алгоритм проведения:

- Открыть веб-приложение
- Выбрать место хранения для удаления из списка
- подтвердить удаление

Ожидаемый результат: успешное удаление места хранения из списка хранилищ.

3.4.4 Тестирование специй

Тест А16
Цель: Проверка добавления специи
Тип: позитивный
Алгоритм проведения:

- Открыть веб-приложение
- Открыть вкладку “Специи”
- Добавить специю, заполнив поля:
    - “название” используя не более 255 символов
    - “бренд” - выбрав позицию из выпадающего списка
    - “место хранения” - выбрав позицию из выпадающего списка

Ожидаемый результат: добавление новой специи в список специй.

Тест А17
Цель: Проверка добавления специи
Тип: негативный
Алгоритм проведения:

- Открыть веб-приложение
- Открыть вкладку “Специи”
- Добавить специю, заполнив поля:
    - “название” используя более 255 символов
    - “бренд” - выбрав позицию из выпадающего списка
    - “место хранения” - выбрав позицию из выпадающего списка

Ожидаемый результат: ошибка добавления.

Тест А18
Цель: Проверка добавления специи
Тип: позитивный
Алгоритм проведения:

- Открыть веб-приложение
- Открыть вкладку “Специи”
- Добавить специю, заполнив поля:
    - “название” используя не более 255 символов
    - не выбирая других пунктов

Ожидаемый результат: добавление специи, с введенным названием и дефолтными брендом и местом хранения.

Тест А19
Цель: Проверка редактирования специи
Тип: позитивный
Алгоритм проведения:

- Открыть веб-приложение
- Открыть вкладку “Специи”
- Редактировать специю, заполнив поля:
    - “название” используя не более 255 символов
    - “бренд” - выбрав позицию из выпадающего списка
    - “место хранения” - выбрав позицию из выпадающего списка

Ожидаемый результат: изменение названия специи в списке специй.

Тест А20
Цель: Проверка редактирования специи
Тип: негативный
Алгоритм проведения:

- Открыть веб-приложение
- Открыть вкладку “Специи”
- Редактировать специю, заполнив поля:
    - “название” используя более 255 символов
    - не изменяя другие поля
    - “место хранения” - выбрав позицию из выпадающего списка

Ожидаемый результат: ошибка редактирования.

Тест А21
Цель: Проверка редактирования специи
Тип: позитивный
Алгоритм проведения:

- Открыть веб-приложение
- Открыть вкладку “Специи”
- Редактировать специю, заполнив поля:
    - “название” используя не более 255 символов
    - не изменяя другие поля

Ожидаемый результат: изменение названия специи в списке специй, без изменения бренда и места хранения.

Тест А22
Цель: Проверка удаления специи
Тип: позитивный
Алгоритм проведения:

- Открыть веб-приложение
- Открыть вкладку “Специи”
- Выбрать специю для удаления из списка
- подтвердить удаление

Ожидаемый результат: удаление специи из списка специй.

4 Отчет о проведении тестирования

4.1 Блочное тестирование

Тест	Дата	Результат	Номер отчета
Б1	13.12.2020	пройден	--
Б2	13.12.2020	провален	1

4.2 Интеграционное тестирование

Тест	Дата	Результат	Номер отчета
И1	13.12.2020	пройден	--
И2	13.12.2020	пройден	--
И3	13.12.2020	пройден	--
И4	13.12.2020	пройден	--
И5	13.12.2020	провален	2
И6	13.12.2020	провален	3
И7	13.12.2020	пройден	--
И8	13.12.2020	провален	4

4.3 Юзабилити тестирование

Тест	Дата	Результат	Номер отчета
Ю1	13.12.2020	пройден	--
Ю2	13.12.2020	пройден	--
Ю3	13.12.2020	пройден	--
Ю4	13.12.2020	пройден	--
Ю5	13.12.2020	пройден	--
Ю6	13.12.2020	пройден	--
Ю7	13.12.2020	пройден	--
Ю8	13.12.2020	пройден	--
Ю9	13.12.2020	пройден	--
Ю10	13.12.2020	пройден	--

4.4 Специальное тестирование

Тест	Дата	Результат	Номер отчета
С1	--	пройден	--

C2	--	пройден	--
C3	--	пройден	--
C4	--	пройден	--

4.5 Аттестационное тестирование

Тест	Дата	Результат	Номер отчета
A1	13.12.2020	пройден	--
A2	13.12.2020	пройден	--
A3	13.12.2020	пройден	--
A4	13.12.2020	пройден	--
A5	13.12.2020	пройден	--
A6	13.12.2020	пройден	--
A7	13.12.2020	пройден	--
A8	13.12.2020	пройден	--
A9	13.12.2020	пройден	--
A10	13.12.2020	пройден	--
A11	13.12.2020	пройден	--
A12	13.12.2020	пройден	--
A13	13.12.2020	пройден	--
A14	13.12.2020	пройден	--
A15	13.12.2020	пройден	--
A16	13.12.2020	пройден	--
A17	13.12.2020	пройден	--
A18	13.12.2020	пройден	--
A19	13.12.2020	пройден	--
A20	13.12.2020	пройден	--
A21	13.12.2020	пройден	--

A22	13.12.2020	пройден	--

5 Журнал найденных ошибок

Отчет об ошибке №1

Тестируемая функция: DatabaseManager.deleteTag()
Краткое описание: тест проверяет, что произойдет ошибка удаления несуществующего тега, вернется соответствующий статус.
Ожидаемый результат: Вернется статус 400, завершение работы функции.
Фактический результат: Вернулся статус 400.
Аномалия: Ошибка поведения программы, соединение после выполнения функции удаления не было закрыто.

Отчет об ошибке №2

Краткое описание: тест проверяет, что произойдет ошибка редактирования несуществующего тега и вернется соответствующий статус.
Ожидаемый результат: Вернется статус 400.
Фактический результат: TypeError: Cannot read property 'tag_id' of undefined at Query.<anonymous> (models/tag-model.js:38:44)

Отчет об ошибке №3

Краткое описание: тест проверяет, что произойдет ошибка редактирования существующего тега при сохранении пустого названия и вернется соответствующий статус.
Ожидаемый результат: Вернется статус 400.
Фактический результат: expect(received).toBe(expected) // Object.is equality Expected: 400 Received: 200

Отчет об ошибке №4

Краткое описание: тест проверяет, что произойдет ошибка удаления не уществующего тега и вернется соответствующий статус.
Ожидаемый результат: Вернется статус 400.
Фактический результат: expect(received).toBe(expected) // Object.is equality Expected: 400 Received: 200

6 Примеры тестов и заглушки

6.1 Пример блочного теста

Библиотека для тестирования: Jest (https://jestjs.io/ru/)
Каждый модуль DatabaseManager представлен отдельным классом.
Код асинхронный, для тестов необходимо прописывать ожидание завершения.

Заглушки:

Класс, имитирующий работу главного файла DatabaseManager - конфигурация для базы данных и соединение с базой данных:

```
class MockDatabaseManager {
  #connection
  constructor(queryResults) {
    this.connection = new MockConnection(queryResults)
  }
  getConnection = () => {
    return new Promise((resolve, reject) => {
      resolve(this.connection);
    });
  }
  showQueries = () => {
    return this.connection.showQueries()
  }
  stop = () => {
  }
}
```

Класс, имитирующий запросы к базе данных:

```
class MockConnection {
  #queries
  #queryResults
  constructor(queryResults) {
    this.queries = []
    this.queryResults = queryResults
  }
  query = (sql, callback) => {
    this.queries.push(sql)
    var result = this.queryResults.shift()
    callback(result.err, result.results, result.fields)
  }
  showQueries = () => {
    return this.queries
  }
  release = jest.fn()
}
```

Пример теста: удаление существующего тега (Б2):

```
test('Delete a tag', async (done) => {
    function callback(data) {
        try {
            expect(data).toBe(1);
        } catch (error) {
            done(error);
        }
    }
    var dbm = new MockDatabaseManager([
        {err: false, results: [], fields: []},
        {err: false, results: [], fields: []}
    ])
    var TagModel = require('../../models/tag-model.js')
    var tagModel = new TagModel(dbm);
    await tagModel.deleteTag(18, callback);
    expect(dbm.showQueries().length).toBe(2);
    expect(dbm.showQueries()[0]).toBe("delete from spice_tag where tag_id='18';");
    expect(dbm.connection.release).toHaveBeenCalled();
    done();
});
```

6.2 Пример интеграционного теста

Библиотеки для тестирования: Jest (https://jestjs.io/ru/) **и Supertest** (https://www.npmjs.com/package/supertest)

Используется тестовая база данных

До каждого теста создается заново база данных с тестовыми данными

```
beforeEach( async (done) =>{
    const dumpFile = './tests/integration/dump_spice.sql';
    execSync(`mysql -u${user} -p${password} -h${host} ${database} < ${dumpFile}`);
    done();
})
```

После каждого теста уничтожается база данных с тестовыми двнными

```
afterEach( async (done) =>{
    const dumpFile = './tests/integration/drop_spice.sql';
    execSync(`mysql -u${user} -p${password} -h${host} ${database} < ${dumpFile}`);
    done();
})
```

Пример теста: взаимодействие Controllers и DatabaseManager, получение списка всех тегов (И3)

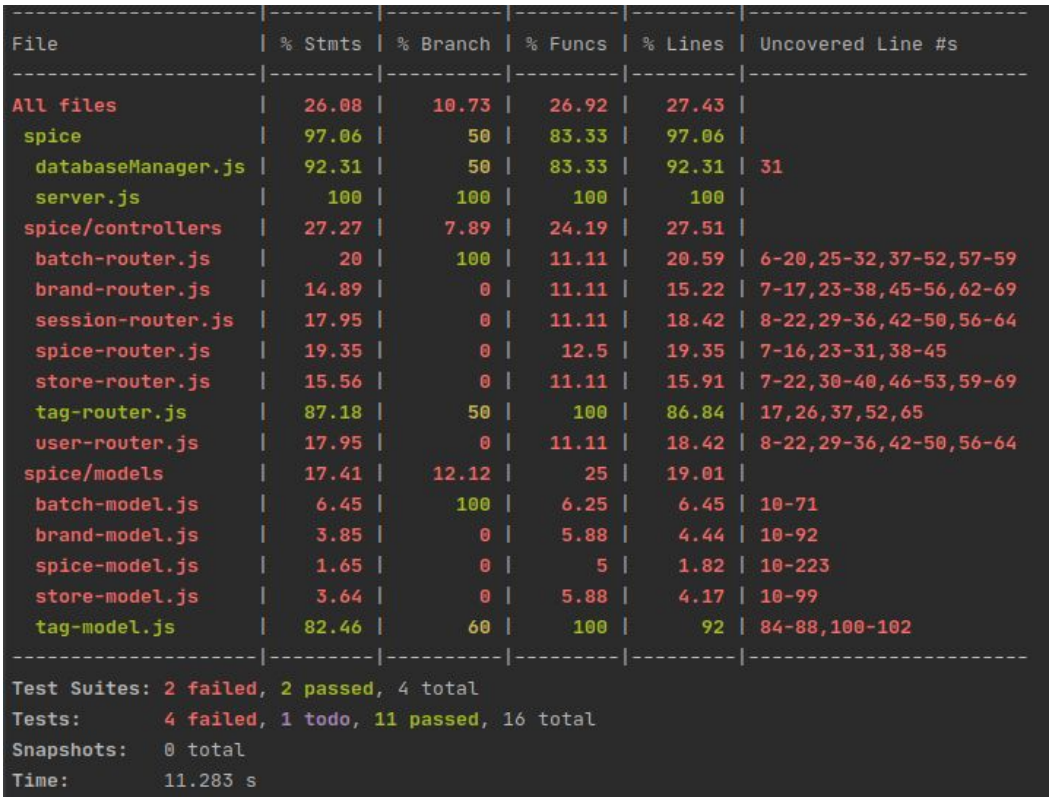
```
        it('Get all tags', async done => {
            const expectedResult = [
```

```
        { tag_id: 1, tag_title: 'miaso' },
        { tag_id: 2, tag_title: 'chicken' },
        { tag_id: 3, tag_title: 'fish' },
        { tag_id: 5, tag_title: 'fffnc' },
        { tag_id: 6, tag_title: 'ggg' },
        { tag_id: 16, tag_title: 'мясо' },
        { tag_id: 17, tag_title: 'макароны' },
        { tag_id: 18, tag_title: 'плов' }
    ];

    const response = await request.get('/api/tags')
    expect(response.status).toBe(200)
    expect(response.body.length).toBe(8)
    console.log(response.body)
    done()
})
```

7 Покрытие кода

Для генерации отчета по покрытия тестами (смотрите изображение ниже) в package.json к подключению jest был добавлен параметр --collectCoverage.



```
----------------------|---------|----------|---------|---------|------------------------
File                  | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s
----------------------|---------|----------|---------|---------|------------------------
All files             |   26.08 |    10.73 |   26.92 |   27.43 |
 spice                |   97.06 |       50 |   83.33 |   97.06 |
  databaseManager.js  |   92.31 |       50 |   83.33 |   92.31 | 31
  server.js           |     100 |      100 |     100 |     100 |
 spice/controllers    |   27.27 |     7.89 |   24.19 |   27.51 |
  batch-router.js     |      20 |      100 |   11.11 |   20.59 | 6-20,25-32,37-52,57-59
  brand-router.js     |   14.89 |        0 |   11.11 |   15.22 | 7-17,23-38,45-56,62-69
  session-router.js   |   17.95 |        0 |   11.11 |   18.42 | 8-22,29-36,42-50,56-64
  spice-router.js     |   19.35 |        0 |    12.5 |   19.35 | 7-16,23-31,38-45
  store-router.js     |   15.56 |        0 |   11.11 |   15.91 | 7-22,30-40,46-53,59-69
  tag-router.js       |   87.18 |       50 |     100 |   86.84 | 17,26,37,52,65
  user-router.js      |   17.95 |        0 |   11.11 |   18.42 | 8-22,29-36,42-50,56-64
 spice/models         |   17.41 |    12.12 |      25 |   19.01 |
  batch-model.js      |    6.45 |      100 |    6.25 |    6.45 | 10-71
  brand-model.js      |    3.85 |        0 |    5.88 |    4.44 | 10-92
  spice-model.js      |    1.65 |        0 |       5 |    1.82 | 10-223
  store-model.js      |    3.64 |        0 |    5.88 |    4.17 | 10-99
  tag-model.js        |   82.46 |       60 |     100 |      92 | 84-88,100-102
----------------------|---------|----------|---------|---------|------------------------
Test Suites: 2 failed, 2 passed, 4 total
Tests:       4 failed, 1 todo, 11 passed, 16 total
Snapshots:   0 total
Time:        11.283 s
```

В столбце Files указаны основные архитектурные блоки приложения.

Критерии покрытия (в отчете приведены в процентах):

- **Stmts** (Statements) - был ли выполнен каждый оператор в программе?
- **Branch** - выполнялась ли каждая ветвь так называемым путем “от решения к решению” в каждой управляющей структуры (например, в операторах if и case)?
- **Funcs** (Functions) - вызывалась ли каждая функция (или подпрограмма) в программе?
- **Lines** - была ли выполнена каждая исполняемая строка в исходном коде? (дополнительно в столбце **uncovered line** указаны номера строк не покрытых тестами)

8 Заключение

Были проведены тесты на удобство использования, нагрузку системы, приемочные тесты, блочно и интеграционно протестировано управление тегами. Обнаруженные ошибки являются критичными и требуют доработки приложения.

В дальнейшем необходимо исправить обнаруженные ошибки и увеличить покрытие тестами, написав интеграционные и блочные тесты для других частей модулей.

Покрытие тестами составило:

- Операторов: 26,08 %
- Ветвей: 10,73 %
- Функций: 26,92 %
- Строк кода: 27,43 %