3 ОТЛАДКА И ТЕСТИРОВАНИЕ ПРОДУКТА

3.1 Тестирование серверной части продукта

В процессе разработки тестирование является неотъемлемой частью работы. Тестирование программного обеспечения – это:

– процесс исследования ПО с целью получения информации о качестве продукта;

– процесс проверки соответствия заявленных к продукту требований и реально реализованной функциональности, осуществляемый путем наблюдения за его работой в искусственно созданных ситуациях и на ограниченном наборе тестов, выбранных определенным образом;

– оценка системы с тем, чтобы найти различия между тем, какой система должна быть и какой она есть.

В широком смысле, тестирование – это одна из техник контроля качества (Quality Control), которая включает планирование, составление тестов, непосредственно выполнение тестирования и анализ полученных результатов.

Важно понимать, что тестирование ПО включает не только собственно проведение тестов, но и многие другие действия, связанные с процессом обеспечения качества:

– анализ и планирование;

– разработку тестовых сценариев;

– оценку критериев окончания тестирования;

– написание отчетов;

– рецензирование документации (в том числе и исходного кода);

– проведение статического анализа.

Виды тестирования: функциональная группа, нефункциональная группа.

Функциональная группа: функциональное тестирование (позитивное / негативное), тестирование безопасности (Security and Access Control Testing).

Нефункциональная группа: нагрузочное тестирование, стресс-тестирование, тестирование стабильности или надёжности, тестирование выносливости, объёмное тестирование, тестирование установки, тестирование удобства пользования, тестирование на отказ и восстановление, тестирование конфигурации, тестирование совместимости.

Важное место при разработке на Node.js занимает тестирование. И в данном случае гораздо легче воспользоваться имеющимися фреймворками, которые упрощают процесс тестирования. Одним из таких фреймворков является «Mocha». Mocha - это многофункциональный тестовый фреймворк JavaScript, работающий на Node.js и в браузере, что делает асинхронное тестирование простым и увлекательным. Тесты Mocha запускаются последовательно, обеспечивая гибкую и точную отчетность и отображая неперехваченные исключения в правильные тестовые случаи.

На примере программного продукта «United Airlines w3.aero», рассмотрим, несколько видов тестирования: интеграционное тестирование, модульное, позитивное, негативное.

Модульное тестирование, иногда блочное тестирование или юнит-тестирование — процесс в программировании, позволяющий проверить на корректность отдельные модули исходного кода программы, наборы из одного или более программных модулей вместе с соответствующими управляющими данными, процедурами использования и обработки.

На рисунке \* представлен код юнит-теста, который проверяет функцию, отправляющую SQL-запрос в базу данных.

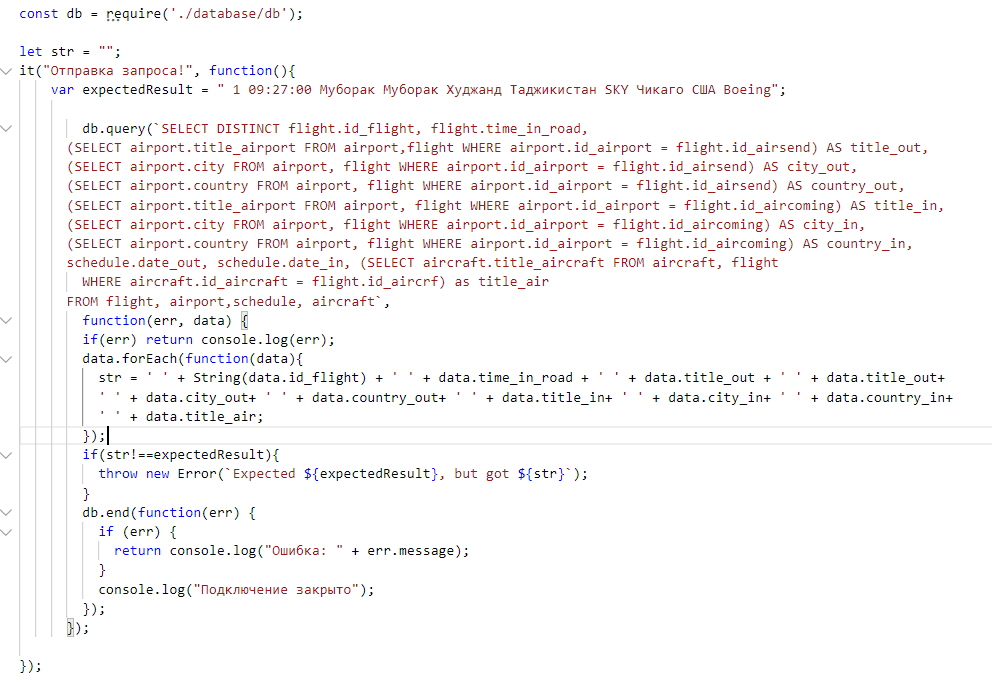


Рисунок \* – код юнит-теста

Интеграционное тестирование – вид тестирования, при котором на соответствие требований проверяется интеграция модулей, их взаимодействие между собой, а также интеграция подсистем в одну общую систему.

На рисунке \* представлен код объединения модулей клиентской части и серверной части, и подключение к базе данных.



Рисунок \* – код объединения модулей клиентской части и серверной части, и подключение к базе данных

Следующий вид тестирования называется «позитивное» тестирование. «Позитивное» – это тестирование на данных или сценариях, которые соответствуют нормальному (штатному, ожидаемому) поведению системы. Как пример, можно рассмотреть бронь билета, в случаи, если все поля входных данных заполнены, бронь пройдет успешно. Основной целью "позитивного" тестирования является проверка того, что при помощи системы можно делать то, для чего она создавалась.

«Негативное» тестирование – это тестирование на данных или сценариях, которые соответствуют нештатному поведению тестируемой системы - различные сообщения об ошибках, исключительные ситуации, "запредельные" состояния и т.п.

Основной целью "негативного" тестирования является проверка устойчивости системы к воздействиям различного рода, валидация неверного набора данных, проверка обработки исключительных ситуаций (как в реализации самих программных алгоритмов, так и в логике бизнес-правил).

Как пример «негативного» тестирования, можно рассмотреть, то, что поля для входных данных не должны быть пустыми, чтобы поставить соответствующее условие достаточно прописать атрибут «required» полям для ввода.

На рисунке \* представлен код и результат негативного тестирования.

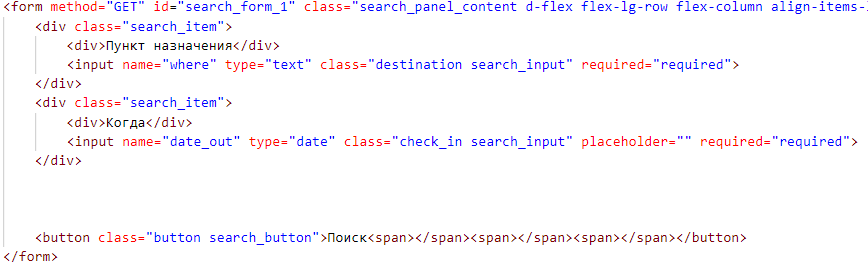


Рисунок \* – результат «негативного тестирования»

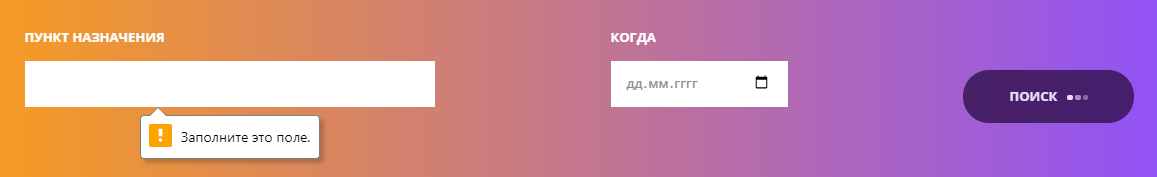


Рисунок \* – результат «негативного тестирования»

3.2 Тестирование клиентской части продукта