5 ПРОГРАММА И МЕТОДИКА ИСПЫТАНИЙ

Для разработки системы использовался такой подход к тестированию, как Test-driven development (TDD). Методология разработки программного обеспечения, при которой разработчик пишет тесты для каждой функции или компонента программы перед тем, как реализовать его код. Эта методология пропагандирует написание тестов перед написанием кода, в отличие от классического подхода к разработке, когда сначала пишется код, а затем тесты на его проверку.

Ближе к концу разработки проводится рефакторинг нового кода для приведения его к соответствующим стандартам. Такой подход является полной противоположностью разработке, при которой сначала разрабатывается программное обеспечение, а затем описываются тестовые ситуации.

Основная идея TDD заключается в том, что писание тестов перед кодом позволяет программисту более точно определить, как должна работать программа, а также позволяет быстро выявлять ошибки и проблемы в процессе разработки. Это помогает повысить качество кода и ускоряет процесс разработки.

В программировании тестирование – это процесс, который определяет, работает ли написанный код правильно. Обычно программисты выполняют тестирование функционала вручную, стимулируя код и проверяя его выполнение. Однако, при автоматическом тестировании компьютер сам стимулирует код и проверяет его работоспособность, что позволяет программисту увидеть результат на дисплее. Это приводит к изменению ответственности, так как теперь от тестов зависит соответствие кода техническому заданию. Методика разработки через тестирование заключается в создании автоматических тестов.

Разработка через тестирование - это методология, в которой разработчик создает автоматизированные модульные тесты, которые определяют требования к коду перед его написанием. Тесты содержат проверки условий, которые могут либо выполниться, либо нет. Когда все условия выполняются, тест считается пройденным, и это подтверждает поведение, которое разработчик ожидает от своего кода. Для создания и автоматизации запуска тестов разработчики используют библиотеки для тестирования. Обычно модульные тесты покрывают критические и нетривиальные участки кода, которые могут быть подвержены частым изменениям, от которых зависит работоспособность другого кода, или которые имеют много зависимостей.

Среда разработки должна быстро реагировать на небольшие модификации кода. Архитектура программы должна базироваться на использовании множества сильно связанных компонентов, которые слабо соединены друг с другом, благодаря чему тестирование кода упрощается.

Техника TDD не только предназначена для проверки правильности работы кода, но также оказывает влияние на дизайн программы. Опираясь на

написанные тесты, разработчики могут лучше понять, какая функциональность необходима для пользователей. Таким образом, детали интерфейса могут быть определены на ранней стадии разработки, до полной реализации решения. Тесты также должны соответствовать стандартам кодирования, применяемым к основному коду.

На рисунке 5.1 приведена схема разработки с применением TDD.

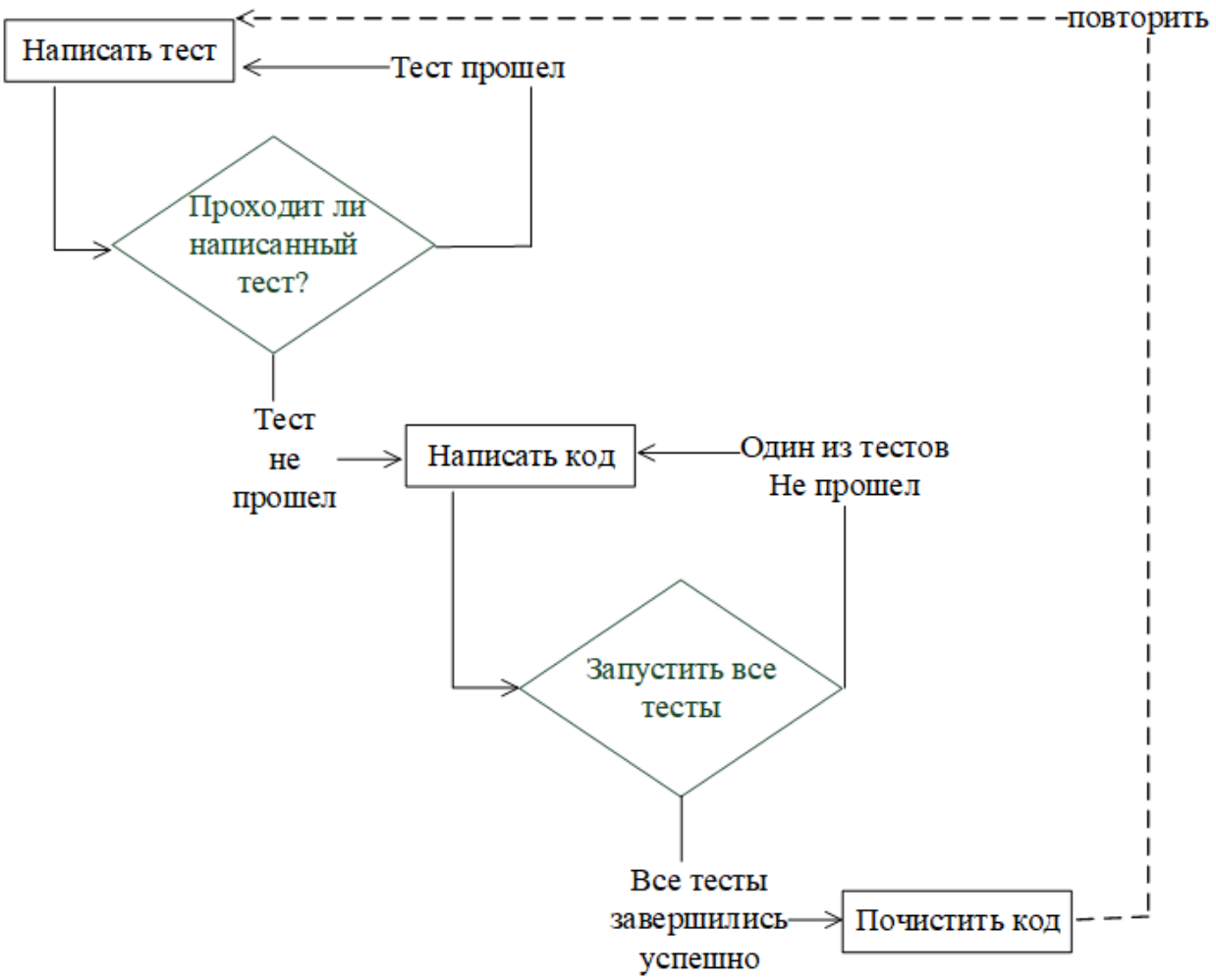

Рисунок 5.1 – Разработка с использованием TDD

В современных проектах значительное внимание уделяется созданию автоматических тестов, поскольку они являются важной частью процесса разработки программного обеспечения. Стандарты разработки, такие как TDD (Test Driven Development) и BDD (Behaviour Driven Development), часто используются для обеспечения высокого покрытия автоматическими тестами. BDD является одним из подходов TDD и может быть реализован в системе RSpec.

RSpec - это фреймворк для написания автоматических тестов на языке программирования Ruby, который предоставляет специальный язык программирования (DSL), называемый спецификацией, для написания тестов.

RSpec используется для практики BDD, которая позволяет описывать желаемое поведение системы на языке бизнес-задач. Спецификация - это отдельный файл, содержащий описание определенной части программы, такой как контроллер, модель, шаблон или хелпер. Файлы спецификаций должны храниться в специальной поддиректории проекта с названием "spec", а имена файлов должны заканчиваться на "_spec.rb".

Иногда возникают ситуации, когда внесение изменений или добавление новых функций может непреднамеренно повлиять на уже существующую функциональность, и разработчик может не заметить эти изменения. Это явление называется регрессией.

Регрессионное тестирование относится к категории тестирования программного обеспечения, которое направлено на выявление возможных ошибок в уже протестированном функционале. Оно используется для обнаружения регрессионных ошибок, которые могут возникнуть при добавлении новых функций или исправлении ошибок, что может привести к появлению новых ошибок в ранее протестированных частях программного кода. Регрессионное тестирование не является средством для доказательства отсутствия ошибок в программе.

Для регрессионного тестирования используются методы, которые включают повторное выполнение предыдущих тестов с целью обнаружения новых регрессионных ошибок, которые могут появиться при добавлении нового функционала. В процессе регрессионного тестирования проводится верификация устранения новых дефектов, проверка того, что дефект, который был исправлен ранее, не возникает снова, и проверка того, что функциональность приложения не нарушена после добавления нового кода и исправления дефектов.

Выделяют несколько видов регрессионных тестов:

-верификационные тесты (проводятся для проверки исправления, обнаруженной ранее ошибки);

- тестирование верификации версии (проверка работоспособности основной функциональности программы).

При написании исходного кода веб-приложения тестирование проводилась в три этапа:

- тестирование каждого отдельного блока в процессе написания программного кода;

- тестирование взаимодействия нескольких блоков между собой;

- полное тестирование программы после окончания процесса написания программного кода.

Эти этапы тестирования являются взаимосвязанными и необходимыми. Модульное тестирование необходимо для выявления проблем в отдельных компонентах программы, а анализ работы программы в целом может быть затруднительным без этого этапа. Однако, работоспособность каждого компонента в отдельности не гарантирует корректного поведения всей программы.

При разработке веб-приложения было проведено модульное тестирование, чтобы выявить возможные проблемы в отдельных компонентах приложения. Также важно отметить проведение сквозного тестирования, которое позволяет проверить работу приложения в целом и выявить возможные проблемы, связанные с взаимодействием различных компонентов приложения.

5.1 Модульное тестирование

Для тестирования серверной части использовалось модульное тестирование. Под подобным тестированием подразумевается процесс, позволяющий проверить отдельные или методы исходного кода.

Основная цель модульного тестирования заключается в том, чтобы писать тесты для каждой функции или метода и проверять все возможные сценарии, к которым можно прийти, как позитивные, так и негативные. Благодаря подобному подходу можно достаточно быстро проверять уже протестированные части кода на регрессии и отслеживать не привело ли последующее изменение к ней. Такой подход облегчает обнаруживать и исправлять подобные ошибки.

5.2 Сквозное тестирование

Сквозное тестирование (end-to-end testing) – это вид тестирования, который проводится на уровне системы в целом. Оно используется для проверки работы программного продукта в условиях, максимально приближенных к реальным. В отличие от модульного тестирования, где проверяется отдельный модуль кода, сквозное тестирование проверяет взаимодействие между компонентами системы в целом.

При сквозном тестировании проводится проверка функциональности всей системы в целом, начиная от пользовательского интерфейса и заканчивая базой данных и взаимодействием с другими системами. На каждом этапе производится проверка входных и выходных данных, а также корректности и своевременности выполнения действий системы.

Сквозное тестирование позволяет выявить не только ошибки в отдельных модулях, но и проблемы взаимодействия между ними. Это может включать ошибки конфигурации, проблемы совместимости, ошибки при передаче данных и другие проблемы, которые могут возникнуть только при взаимодействии компонентов системы в целом.