[3 Разработка технологии тестирования 2](#_Toc1)

[3.1 Выбор подходов и методов тестирования 2](#_Toc2)

[3.2 Автономное тестирование компонентов ПО 3](#_Toc3)

Обозначения

ПО — ...

API — ...

# 3 Разработка технологии тестирования

## **3.1 Выбор подходов и методов тестирования**

Процесс разработки программного обеспечения в том виде, как оно определяется в современной модели жизненного цикла программного обеспечения предполагает три стадии тестирования [1]:

* автономное тестирование компонентов ПО;
* комплексное тестирование разрабатываемого ПО;
* системное или оценочное тестирование на соответствие основным критериям качества.

Для проведения автономного и комплексного тестирования необходимо сформировать тестовые наборы, опираясь на структурный или функциональный подход.

Структурный подход базируется на том, что известна структура тестируемого ПО, в том числе его алгоритмы. Тесты строят так, чтобы обеспечить максимальное покрытие исходного кода.

Функциональный подход основывается на том, что структура ПО не известна. В этом случае тесты строят, опираясь на функциональные спецификации. Тесты строят на базе различных способов декомпозиции множества данных.

Разработанное ПО включает в себя разнородные алгоритмы, для всестороннего тестирования которых с помощью структурного подхода понадобились бы значительные затраты времени на изучение исходного кода, разработку большого числа тестов и заглушек. По этой причине было решено использовать функциональный подход, который позволил бы значительно сократить время на разработку тестов, обеспечивая при это тестирование всей необходимой функциональности [2].

В качестве оценочного тестирования согласно ТЗ было выбрано нагрузочное тестирование.

## 3.2 Автономное тестирование компонентов ПО

Так как написание тестов на Golang требует значительного времени и такие тесты сложнее поддерживать в силу непопулярности языка среди тестировщиков, было решено тестировать разработанные микросервисы, предварительно запустив их (см. приложение Х) и обращаясь к ним по протоколу HTTP. Такой подход позволил реализовать тесты не привязываясь к языку реализации исходного ПО.

Поскольку Python обладает простым синтаксисом, большим количеством библиотек и популярен среди тестировщиков (рисунок Х), именно он был выбран для реализации тестов [3].



Рисунок Х — наиболее популярные языки в области автоматизированного тестирования

## В качестве основной библиотеки для тестирования была выбрана библиотека pytest, являющаяся одной из наиболее популярных библиотек для автоматизированного тестирования [4].

## Pytest обладает следующими основными преимуществами [5]:

* меньше повторяющегося кода за счет независимости от API;
* выполнение определенного набора тестов с помощью фильтрации;
* параметризация тестов — запуск одного и того же теста с разными наборами параметров;
* гибкость — архитектура библиотеки основана на плагинах, которые можно установить отдельно;
* полная обратная совместимость с unittest — возможность запуска тестов, написанных на нем;
* выполнение нескольких тестов параллельно;
* установочный код можно использовать повторно.

## В дополнение к pytest была использована библиотека allure, формирующая интерактивные отчеты о прохождении тестов. Тесты в allure можно иерархически группировать и сопровождать логами и вложениями. Allure поддерживается не только для Python, но и для Java, JavaScript, Ruby, PHP , .Net и Scala. Такой широкий набор поддерживаемых языков программирования делает allure (рисунок Х) знакомым многим разработчикам, тестировщикам и менеджерам, что упрощает поддержку тестов.



## Рисунок Х — интерфейс allure

Источники

1. Книга Ивановой
2. <https://coderlessons.com/tutorials/kachestvo-programmnogo-obespecheniia/ruchnoe-testirovanie/chernyi-iashchik-protiv-belaia-korobka>
3. <https://habr.com/ru/post/543346/>
4. https://www.softwaretestinghelp.com/python-testing-frameworks/