Создание таблицы со сравнительным анализом библиотек для тестирования. Формирование отчета по выполнению задания и размещение его в портфолио, персональном репозитории.

|  |  |  |
| --- | --- | --- |
| Библиотека | Плюсы | Минусы |
| PyTest | Главное достоинство конфигурации — легкость ее использования, расширяемость и возможность быстрого изменения структуры автоматизированных тестов. Можно оперативно настроить систему отчетов, например, подключить ReportPortal или аналогичный инструмент для их формирования.  Также у PyTest есть фикстуры — это, по сути, декораторы в Python. С их помощью можно делать setup и teardown на разных уровнях (“function”, “class”, “module”, “session”), параметризацию, установку меток. | В целом найти серьезные минусы у этой конфигурации практически невозможно: это проверенный временем подход, который отлично себя показывает на любых проектах. Из относительных недостатков можно назвать необходимость знания Python на высоком уровне. Правда, в современных реалиях без языка программирования в автоматизации тестирования делать нечего, поэтому назвать это минусом можно лишь формально. |
| Robot Framework | Основной плюс конфигурации Python + Robot Framework — это возможность писать автоматизированные тесты на кейвордах силами сотрудников без знания языка программирования. Внутри фреймворка реализована часть функций «из коробки», и для их использования обладать навыками работы с кодом не нужно. Но на деле преимущество подхода оборачивается самообманом — подробнее об этом рассказано в минусах.  Также в самом фреймворке есть неплохая система логирования, редактор тестов, множество дополнительных плагинов для интеграции в проекты. Архитектура выстроена так, что инженер может расширять функциональность и писать свои библиотеки на Python. | Без детального анализа и погружения в код понять, что упало внутри Python, вызванного из Robot Framework, невозможно. Могут появиться дополнительные сложности с дебагом кода автотестов, с которыми без знания программирования не справиться. Если меняется продукт или инфраструктура, то тестировщику все равно понадобится помощь специалиста, который знает язык и умеет программировать, чтобы исправить все «под капотом» приложения.  Автотесты при таком подходе работают медленнее из-за недостатков фреймворков и создания дополнительного уровня абстракции.  Разработка и поддержка кода функций «под капотом» проекта часто занимает больше времени, чем мы экономим при написании тестов на естественном языке. |
| Behave | Те же, что и у Robot Framework — возможность писать автоматизированные тесты силами специалистов без знания языка программирования. И минус этого фреймворка характерен для Behave: без умения погружаться в код и понимания Python писать тесты будет невозможно или очень сложно, трудоемко и в результате неэффективно. Однако некое подобие фикстур из PyTest позволяет удобно модифицировать поведение функции, не изменяя ее код.  Кроме того, Behave легко встраивается в серверы непрерывной интеграции и формирует красивые и понятные отчеты. | Минусы Robot Framework правомерны и для Behave, но есть и другие особенности. Например, отсутствует возможность распараллелить автотесты, благодаря которой многократно уменьшается время их прохождения. Несмотря на то, что далеко не всё нужно распараллеливать, сама возможность это делать критически важна для серьезной автоматизации.  Разработчики фреймворка обещают добавить ее, но пока при попытке распараллелить автоматизированные тесты данные в сущности context «липнут» друг на друга. |