4.1. Создание таблицы со сравнительным анализом библиотек для тестирования.

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Описание | Преимущества | Недостатки |
| Robot Framework (RF) | Robot Framework (RF) – открытый фреймворк автоматизации тестирования для приемочного тестирования, ATDD и RPA. Его ядро написано на Python, но может запускаться на Jython (Java-реализация Python) и IronPython (Python для .NET framework). Для начала работы вам потребуется установить Python версии 2.7.14 или выше. | Основывается на подходе Keyword-driven testing (KDT), что позволяет нам легко создавать тест-кейсы с использованием удобных для восприятия человеком ключевых слова (не требует опыта написания кода).  Поддерживает все операционные системы (Windows, Linux или macOS) и все приложения (веб, мобильные и десктопные приложения).  Предоставляет понятные и удобные данные HTML-отчетов (включая скриншоты).  Богатая экосистема с множеством API, что делает его хорошо расширяемым фреймворком и позволяет интегрироваться любым другим сторонним инструментом.  Поддерживает синтаксис If/Else, начиная с RF v4.0.  Отлично поддерживается сообществом и имеет множество онлайн-ресурсов.  Основывается на подходе Keyword-driven testing (KDT), что позволяет нам легко создавать тест-кейсы с использованием удобных для восприятия человеком ключевых слова (не требует опыта написания кода).  Поддерживает все операционные системы (Windows, Linux или macOS) и все приложения (веб, мобильные и десктопные приложения).  Предоставляет понятные и удобные данные HTML-отчетов (включая скриншоты).  Богатая экосистема с множеством API, что делает его хорошо расширяемым фреймворком и позволяет интегрироваться любым другим сторонним инструментом.  Поддерживает синтаксис If/Else, начиная с RF v4.0.  Отлично поддерживается сообществом и имеет множество онлайн-ресурсов. | Из коробки не поддерживается параллельное тестирование, но его можно использовать с помощью Selenium Grid (https://www.selenium.dev/documentation/en/grid/) или через Pabot (https://pabot.org/) (параллельный исполнитель для RF).  Хорошо это или плохо, но он заставляет вас работать в соответствии с заранее определенной методологией, поэтому поначалу кривая обучения может быть длиннее, чем обычно.  Создание ключевых слов-дженериков может занять больше времени, чем обычное написание кода.  Сложно настроить отчеты. |
| Pytest | Pytest – это открытый фреймворк для тестирования, который является одним из широкоиспользуемых в Python. Pytest также поддерживает модульное, функциональное и тестирование API. Для его запуска вам понадобится Python версии 3.5 или выше. | Позволяет создавать компактные и простые наборы тестов.  Легко расширяется с помощью таких плагинов как: pytest-randomly, pytest-cov, pytest-django, pytest-bdd.  Еще вы можете добавить в свой проект плагин pytest html для печати HTML-отчетов с помощью одного просто флага в командной строке.  Можно выполнять тесты параллельно с помощью плагина pytest-xdist. Больше о нем можно узнать здесь.  Имеет очень большое сообщество.  Поддерживает фикстуры, позволяющие охватить все комбинации параметров без надобности переписывать тест-кейсы. Также они являются отличным способом управлять контекстом между этапами. | Совместимость – не ключевой фактор для Pytest, поскольку вы хоть и можете с легкостью писать тест-кейсы на Pytest, но не можете использовать их в другом фреймворке из-за использования уникальных для Pytest процедур. |
| TestProject | TestProject – это полностью бесплатный фреймворк для автоматизированного тестирования с облачными и локальными HTML-отчетами. С помощью TestProject вы можете с легкостью собрать автоматизацию для тестирования мобильных, веб или иных приложений с помощью открытого SDK. Он поддерживает Python версии 3.6 и выше, а также фреймворки Pytest и Unittest (узнать больше можете здесь). TestProject включает в себя все зависимости, необходимые для обеспечения единственного кроссплатформенного тестирующего агента (посмотрите этот вебинар, чтобы узнать больше). | Агент, включающий все сторонние библиотеки, необходимые для выполнения и разработки автоматизации тестирования для мобильных, веб или generic-тестов.  Бесплатные автоматические отчеты в форматах HTML/PDF (со скриншотами).  История выполнения доступна через RESTful API.  Всегда актуален за счет последних и стабильных версий драйверов Selenium/Appium.  Единый SDK для Web, Android, iOS и Generic-тестов.  Встроенный раннер и функционал для составления отчетов.  Кроссплатформенная поддержка для Mac, Windows, Linux и Docker.  Большое сообщество и поддержка: форум, блог и встроенный чат. | Агент выполняет по одному тесту за раз, поэтому для параллельного тестирования вам понадобятся агенты Docker.  Функции коллаборации из гибридного облака ограничены при работе в автономном режиме. То есть в отличие от бесшовной совместной работы в гибридном облаке, при использовании локальной опции «on-prem» вам нужно будет реализовать совместную работу самостоятельно, сохранив тесты на общем сетевом диске/git. |
| PyUnit (Unittest) | PyUnit (Unittest) – фреймворк для модульного тестирования на Python, вдохновлённый JUnit. Является частью стандартной библиотеки Python, таким образом, большинство разработчиков начинают свой путь в тестирование с него. | Поскольку этот фреймворк является частью стандартной библиотеки, никаких дополнительных модулей ставить не нужно – все поставляется из коробки.  Предлагает простое и гибкое выполнение тест-кейсов.  Быстрая генерация отчетов о тестах в XML и unittest-sml-reporting. | Цель кода теста порой становится неясной ввиду поддержки абстракции.  Большое количество шаблонного кода.  Из-за того, что фреймворк основан на Junit, для именования используется camelCase, а не snake\_case, характерный для Python. |
| Nose2 | Nose2 – это преемник Nose, который основан на PyUnit (Unittest), но с плагинами. Nose2 расширяет возможности PyUnit с помощью различных плагинов, которые добавляют поддержку выполнения тестов, обнаружение тестов, декораторы, фиктуры, параметризацию и т.д. | Легко начать, поскольку он расширяет фреймворк PyUnit (Unittest), который поставляется из коробки в стандартной библиотеке Python.  Включает в себя большое количество встроенных плагинов, которые могут упростить и ускорить тестирование.  Поддерживает параллельное тестирование с помощью плагина mp.  Автоматически собирает тесты, если вы следуете рекомендациям по организации своей библиотеки и кода тестов. | Отсутствие богатой документации, что может затормозить вас в самом начале.  Не так активно поддерживается в отличие от других фреймворков. |
| Behave | Behave – один из самых популярных фреймворков BDD-тестирования (behavior-driven development) на Python. Несмотря на то, что он официально не является частью проекта Cucumber, он функционирует очень схоже с фреймворками Cucumber. | Позволяет писать тест-кейсы на удобочитаемом языке, что упрощает совместную работу между командами с аналогичным функционалом.  Имеется большой объем документации и поддержка, которая поможет вам начать работу.  Полностью поддерживает Gherkin, таким образом создание функциональных файлов не требует особых технических знаний.  Есть интеграция с Django и Flask. | Нет поддержки параллельного выполнения.  Только black-box тестирование. |
| Lettuce | Lettuce – другой BDD-фреймворк Python, основанный на Cucumber. Требует Python 2.7.14 или выше. | Поддерживает Gherkin, что позволяет членам команды без технического бэкграунда легко создавать тесты на естественном языке.  По аналогии с Behave, он в основном для black-box тестирования, но может использоваться и для большего количества видов тестирования. Например, Lettuce может тестировать различные модели поведения и взаимодействия серверов и баз данных. | Не хватает некоторого функционала из других фреймворков, поэтому Lettuce больше подходит для небольших проектов.  Не похоже, чтобы сам фреймворк и его документация активно поддерживались.  Чтобы убедиться в успешности реализации, нужно установить особую коммуникацию между всеми стейкхолдерами в проекте: разработчиками, тестировщиками и менеджерами. |
| Testify | Testify создан на замену Unittest и Nose и обладает расширенным функционалом по сравнению со стандартным Unittest. | Используется для модульного, интеграционного и системного тестирования.  Для тех, кто знаком с Unittest, Testify очень прост в работе.  Наличие большого количества плагинов.  Подобно Nose2, Testify умеет обнаруживать тесты.  Простой синтаксис фикстур. | Отсутствие большой документации, поэтому новичкам придется искать многие вещи самостоятельно.  Сложная реализация параллельного тестирования. |