Стек:

pyenv => Selenium, requests

Дополнительно:

pyenv – возможность безвозмедного выбора нужной версии python

pipenv – наиболее акутальная система контроля библиотек для работы с продуктом.

pytest – интсрумент автотестов

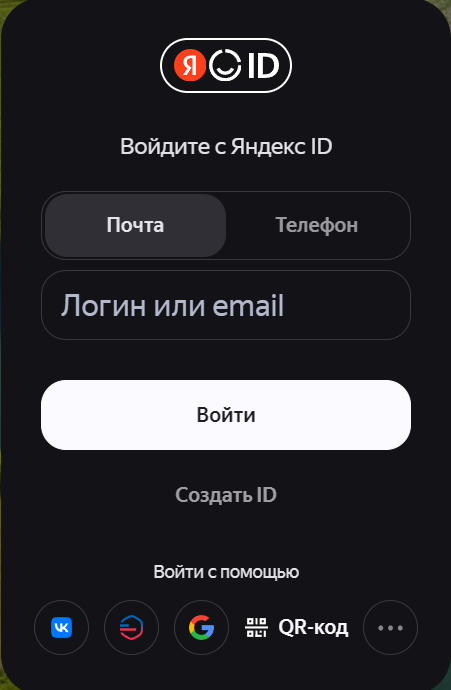
Selenium – вебдрайвер для тестирования веб

requests – библиотека для отправки/получения запросов в API

## Архитектура проекта:

1. suites *-* *директория сьютов с вызовами тестов из framework.tests*
   1. Suite\_A
      1. test\_A\_1.py
      2. test\_A\_2.py
   2. Suite\_B
      1. test\_B\_1.py
      2. test\_B\_2.py
2. framework *–* *директория тестов и методов*
   1. functionality *–* *функциональные компоненты автотестов*
   2. tests *–* *наборы тестов для каждой из страниц*
      1. test\_login\_page.py
3. page\_objects *–* *директория с объектами страницы*
   1. base *–* *директория с базовыми надстройками для Page Objects и WebElements*
      1. base\_page.py
      2. base\_element.py
   2. page\_1.py
   3. page\_2.py
4. config – *директория конфигурационных файлов*
5. reports – *директория отчетов со скриншотами*

## Объект тестирования



Объектом тестирования является окно авторизации в Яндекс ID.

## Тест-кейсы

Всего создано 14 тест-кейсов, 5 из которых – с высоким приоритетом

Тест-кейсы лежат внутри репозитория git.

# Архитектура кода позволяет:

* Создавать отдельные тест-кейсы под каждую веб-страницу путем PageObjects.
* Применять и дополнять методы взаимодействия со страницей (refresh, previous page, etc) и ее веб-элементами (click, input\_text, clear, etc.)
* Создавать функциональные методы, применяющие базовые, либо дополняющие фреймворк (директория /framework/functionality), например парсинг информации через requests, создание таблиц в реляционной SQLite или запросом в API для получения данных (например, код СМС)
* Для удобства создаются отдельные сьюты в которых создаются кортокие скрипты для запуска тестов из основных модулей (framework/tests) создающиеся под отдельные страницы (например, тесты в login\_page\_tests.py, сьюты для запуска скриптов из login\_page\_tests.py – в suites/suite\_LoginPageTests) для которых в дальнейшем можно настроить уровни Severity в качестве отдельных конфиг-файлов (не сделано)
* Не требуется скачивание webdriver\_chrome. Это происходит автоматически благодаря библиотеке webdriver\_manager.

Недочеты:

1. **Сильный недочет.** Невозможна параметризация через скрипты в suites в силу криво написанного декоратора внутри BaseTest (@BaseTest.test\_case), который нужен для связи тестов с системой логгирования и закрытия веб-драйвера. Также в идеале хотелось сделать, чтобы передавался номер тест-кейса для связи логов, например, с Jira. Ситуация решается, но я не придумал как это сделать.
2. **Незначительный недочет.** Не применяется pytest, хотя в случае этого фреймворка можно дописать собственными модулями. Поэтому, считаю, не такой сильный минус.
3. **Средний недочет.** Тестирование происходит только с Chrome. Но автоматизация возможна и под GeckoDriver или Edge. Для этого надо параметризовать сам BaseTest либо @BaseTest.test\_case. Для Gecko потребуется настроить webdriver\_manager.
4. **Незначительный недочет.** На текущий момент отчеты выгружаются в виде файлов txt. Также именование несколько кривое, поэтому информацию нужно уточнять внутри самого файла, в котором указан какой тест был произведён.