ФИЛОСОФИЯ.ИТ
РОСАТОМ

Общество с ограниченной ответственностью «Философия.ИТ»
Россия, 107023, г. Москва, ул. Измайловский Вал, д. 30
Тел.: +7 (495) 988-37-38, факс: +7 (495) 988-37-38,
e-mail: customer@fil-it.ru
ИНН 7713728490
КПП 771901001
ОГРН 1117746379145

РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

УНИВЕРСАЛЬНОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ДЛЯ АВТОМАТИЗИРОВАННОГО ТЕСТИРОВАНИЯ ЦИФРОВЫХ СИСТЕМ «ТАЙМЫР.АВТОТЕСТИРОВАНИЕ» (TAYMYR.AUTOTEST)

г. Москва

Оглавление

1. Общие положения

1.1. Полное наименование программы для ЭВМ, обозначение

Полное наименование Программы для ЭВМ: «Универсальное программное обеспечение для автоматизированного тестирования цифровых систем "Таймыр.Автотестирование" (Taymyr.Autotest)» – далее по тексту Система.
Краткое наименование (обозначение) Системы: Таймыр.Автотестирование (Taymyr.Autotest)

1.2. Назначение системы

Таймыр.Автотестирование представляет собой систему для автоматизированного тестирования цифровых платформ. Зоны охвата автоматизации: Web, API, мобильные приложения. ПО предназначено для реализации и запуска автоматизированных тестов, а также формирования отчетности по результатам запуска. ПО служит для уменьшения трудозатрат тестирования в цикле разработки.

1.3. Состав системы

Система состоит из 4 частей:

- Модуль для автоматизации тестирования Web. Предназначен для написания автотестов, которые делают проверки в Web-интерфейсе.
- Модуль для автоматизации тестирования API. Предназначен для написания автотестов, которые делают проверки API.
- Модуль для автоматизации тестирования мобильных приложений. Предназначен для написания автотестов, которые делают проверки в мобильных приложениях.
- Общий модуль с реализаций функционала, который используют другие модули.

1.4. Разработчик системы

Полное наименование: Общество с ограниченной ответственностью «Философия.ИТ».
Сокращенное наименование: ООО «Философия.ИТ».

1.5. Назначение документа

Настоящий документ входит в комплект эксплуатационной документации по Таймыр.Автотестирование и содержит руководство пользователя Системы.

2. Описание

Инструмент представляет собой фреймворк в виде исходного кода, размещенного в репозитории.

Весь код разделен на 4 глобальных модуля:

- **api-tests** (содержит реализацию кода, позволяющую писать автотесты на проверку REST API, а также непосредственно пример автотеста);
- **web-tests** (содержит реализацию кода, позволяющую писать автотесты на проверку клиентской части WEB-приложения, а также непосредственно пример автотеста);
- **mobile-tests** (содержит реализацию кода, позволяющую писать автотесты для проверки функций мобильных приложений для iOS и Android, а также непосредственно пример автотеста);
- **common** (содержит общие механизмы для вышеперечисленных модулей, позволяющие производить интеграционные взаимодействия с популярными в IT-ландшафтах системами).

Во фреймворке использованы технологии с открытым исходным кодом, а именно:

- JavaScript и TypeScript – языки программирования.
- BDD-инструмент Cucumber совместно с сценарно-ориентированным языком Gherkin (для легкой читаемости сценариев и переиспользования общих шагов).
- Библиотека для тестирования WEB – WebdriverIO.
- Библиотеки для тестирования Mobile – Appium, WebdriverIO.
- Библиотека для тестирования API – PactumJS.

Запуск фреймфорка может быть произведен локально (через IDE), а также с помощью CI/CD (Jenkins, GitLab CI/CD и другие).

Краткое техническое описание работы фреймворка:

- Написание сценариев производится на сценарно-ориентированном языке Gherkin. В предоставленном коде уже есть все необходимые вариации использования Gherkin, поэтому нет необходимости в дополнительной инструкции для работы с фреймворком.

  По ссылке можно найти исчерпывающую документацию по Gherkin на английском языке. Ниже приведена таблица адаптации ключевых слов в описании сценариев для русского языка (во фреймворке используется русский язык для описания сценариев):

Английские ключевые слова	Русские ключевые слова
feature	Функция Функциональность Функционал

Английские ключевые слова	Русские ключевые слова
	Свойство Фича
background	Предыстория Контекст
scenario	Пример Сценарий
scenarioOutline	Структура сценария Шаблон сценария
examples	Примеры
given	* Допустим Дано Пусть
when	* Когда Если
then	* То Затем Тогда
and	* И К тому же Также
but	* Но А Иначе
rule	Правило

- Реализация шагов сценариев производится на языке программирования TypeScript, при этом одни и те же шаги для разных сценариев могут быть переиспользованы.
- Для описания объектов интерфейса WEB/Mobile используется паттерн PageObject.
- Для проверки работы API используется механизм вариативного сравнения объектов.

- Интеграционные механизмы (с СУБД, брокером сообщений, и т.д.) вынесены в отдельный модуль и используется всеми модулями автотестирования (WEB/Mobile/API).

3. Навигация по коду и алгоритм написания автотестов

3.1. API

Модуль для автотестирования API **api-tests** имеет следующие функциональные и конфигурационные директории:

- features (/api-tests/src/features). В ней логическим способом по структуре распределяются файлы с расширением .feature, в которых на Gherkin описаны сценарии. Рядом с этими файлами в директории с названием data находятся json-файлы с данными для этих сценариев. Визуально структура отображена на **Рисунок 1**.

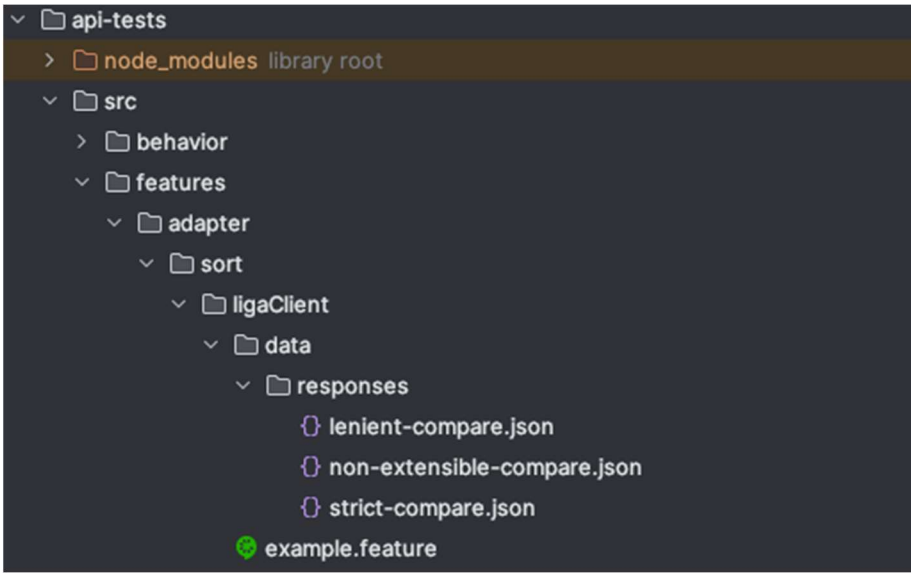

Рисунок 1

- behavior (/api-tests/src/behavior). В ней в соответствующих директориях находится реализация, конфигурация и вспомогательные сервисы:
  - config. Конфигурация, в частности, для отчетности (**Рисунок 2**).

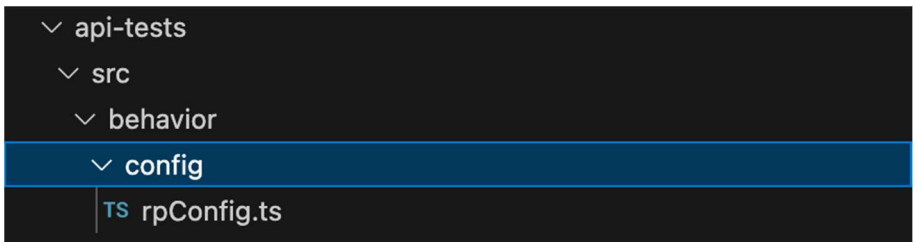

Рисунок 2

- steps (вложенная директория base, вложенная директория hooks). Реализация шагов тестовых сценариев, предусловия и постусловия (**Рисунок 3**, **Рисунок 4**).

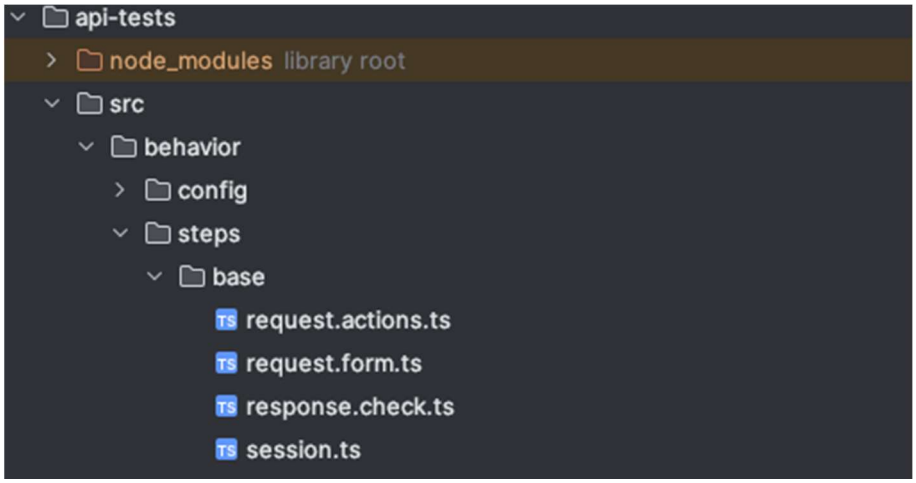

Рисунок 3

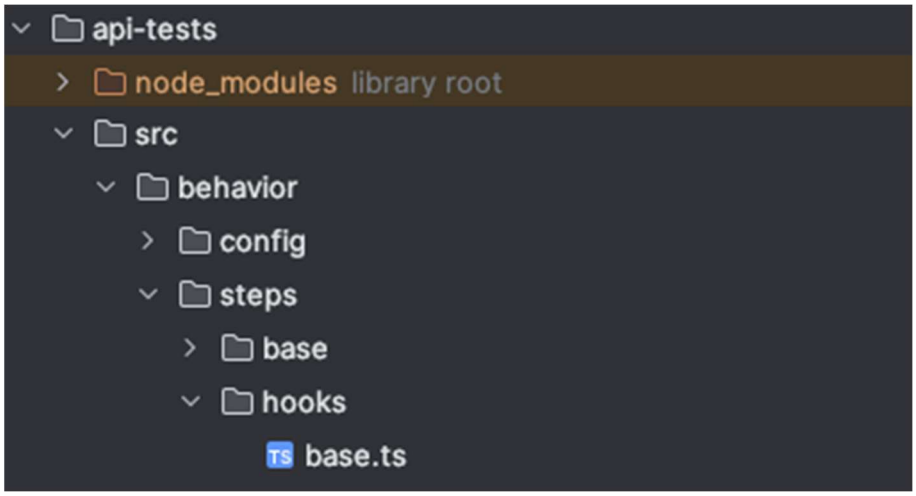

Рисунок 4

В модуле автотестирования API предусмотрено исчерпывающее количество реализованных шагов, поэтому для написания API-автотеста необходимо реализовывать только сценарии на Gherkin, в написании кода необходимости нет (при этом техническая кастомизация кода может быть произведена в случае необходимости). Для написания API-автотеста необходимо:

- В директории /api-test/src/features создать директорию для сценария.
- Внутри создать файл с расширением .feature и написать в нем нужный сценарий, используя готовые шаги и передавая в них нужные параметры.
- На том же уровне, что и файл с расширением .feature, создать директорию data (при необходимости использования данных для теста), внутри создать нужные файлы .json с данными.

Для сравнения ожидаемого и фактического результата (полного json-файла/конкретных параметров ответа) нужно использовать предусмотренную вариативную механику сравнения из **common** (модуль будет описан далее):

- Сравнение json. Шаг “И тело ответа JSON "path/to/file/file.json" будет проверяться по правилу "mode"”). Существуют следующие вариации переменной mode:
    - STRICT – строгое сравнение json друг с другом, два json проверяются на полную идентичность.
    - LENIENT – нестрогое сравнение json. В фактическом json будут проверяться только те параметры, которые ожидаются, “лишние” параметры в фактическом json игнорируются.
    - NON_EXTENSIBLE – используется в случае, когда не важен порядок объектов в массиве, а важно лишь их наличие со всеми ожидаемыми параметрами в ответе.
- Сравнение параметров. Шаг “И значения JSON объектов попадают под следующие ограничения:

locator	restrictionMode	restrictionValue
error	IS_EQUAL_TO_NUMBER	0
content.branches[0].code	IS_EQUAL_TO_STRING	test |”

  Существуют следующие вариации restrictionMode:

    - IS_EQUAL_TO_STRING – сравнение со строкой.
    - IS_EQUAL_TO_CASE_INSENSITIVE – сравнение со строкой без учета регистра.
    - IS_EQUAL_INTEGER – сравнение с целым числом.
    - IS_EQUAL_TO_NUMBER – сравнение с числом.
    - IS_EQUAL_TO_BOOLEAN – сравнение с булевым значением.
    - IS_EMPTY – сравнение с пустым объектом.
    - CONTAINS_ONLY_VALUES – совпадение хотя бы с одним из значений из списка (разделитель значений в списке – ";").
    - NOT_CONTAINS_VALUES – отсутствие совпадений со всеми значениями из списка (разделитель значений в списке – ";").
    - CONTAINS_SUBSTRING – совпадение по подстроке.
    - IS_NOT_BLANK – объект не пустой.
    - IS_EXIST_WITH_VALUES – проверка, что объект принимает одно из указанных значений, либо пустой.

- IS_REGEX - совпадение по регулярному выражению.
- IS_STRING - значение является строкой (сравнивается только тип, а не само значение).

3.2. WEB

Модуль для автотестирования WEB **web-tests** имеет следующие функциональные и конфигурационные директории:

- features (/web-tests/src/features). В ней логическим способом по структуре распределяются файлы с расширением .feature, в которых на Gherkin описаны сценарии. Визуально структура отображена на **Рисунок 5**.

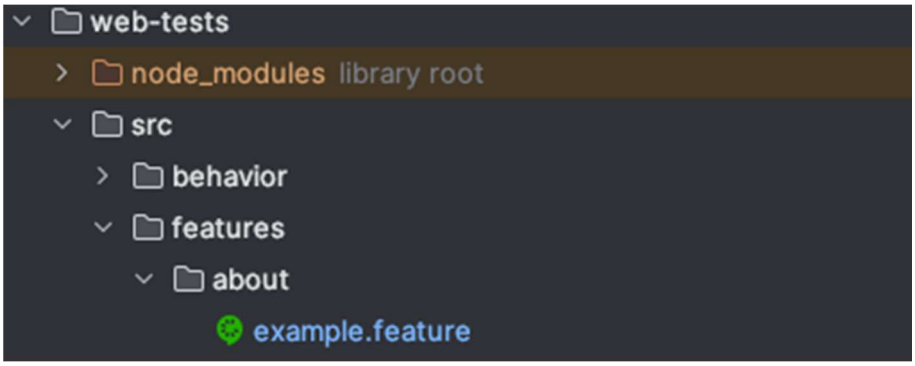

Рисунок 5

- behavior (/web-tests/src/behavior). В ней в соответствующих директориях находится реализация, конфигурация и вспомогательные сервисы:
  - actions. Вспомогательные сервисы для шагов WEB-сценариев (**Рисунок 6**).

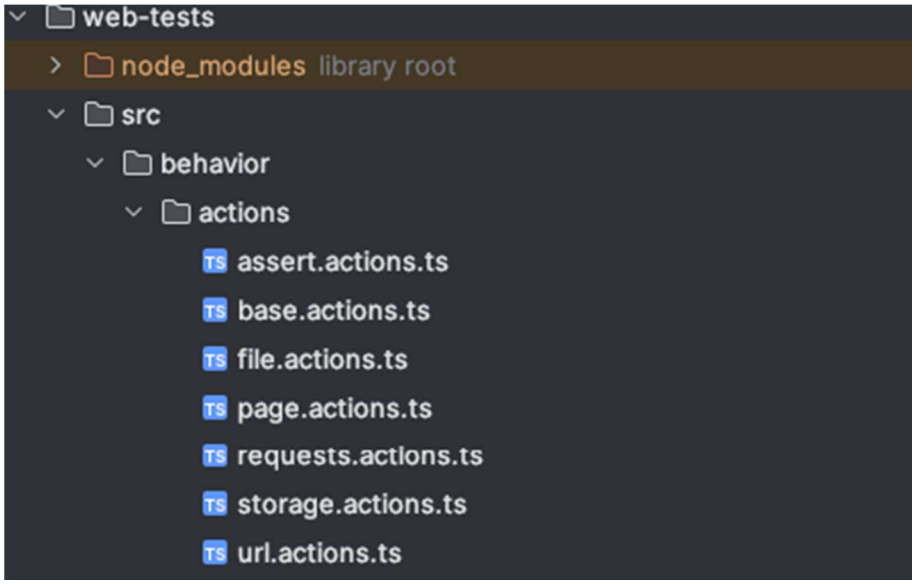

Рисунок 6

- pageobjects. Описание объектов интерфейса страниц (**Рисунок 7**).

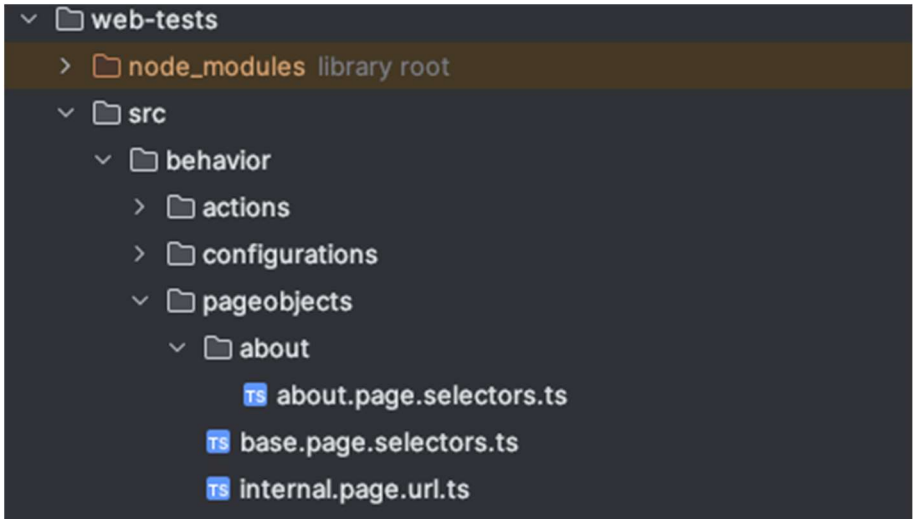

Рисунок 7

- steps (корневой раздел директории, вложенная директория hooks). Реализация шагов тестовых сценариев, предусловия и постусловия (**Рисунок 8**).

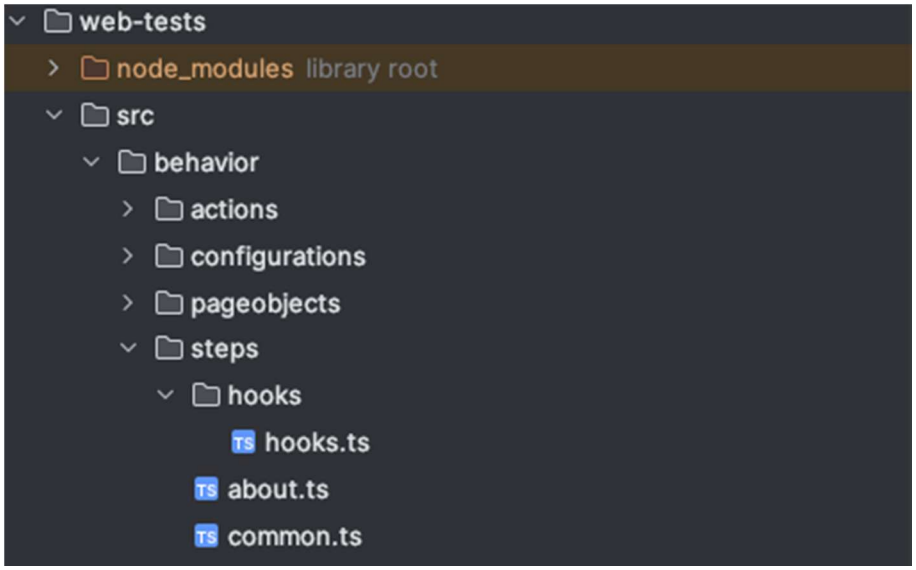

Рисунок 8

- configurations. Конфигурация для отчетности, конфигурация и сервисы для запуска тестов (**Рисунок 9**).

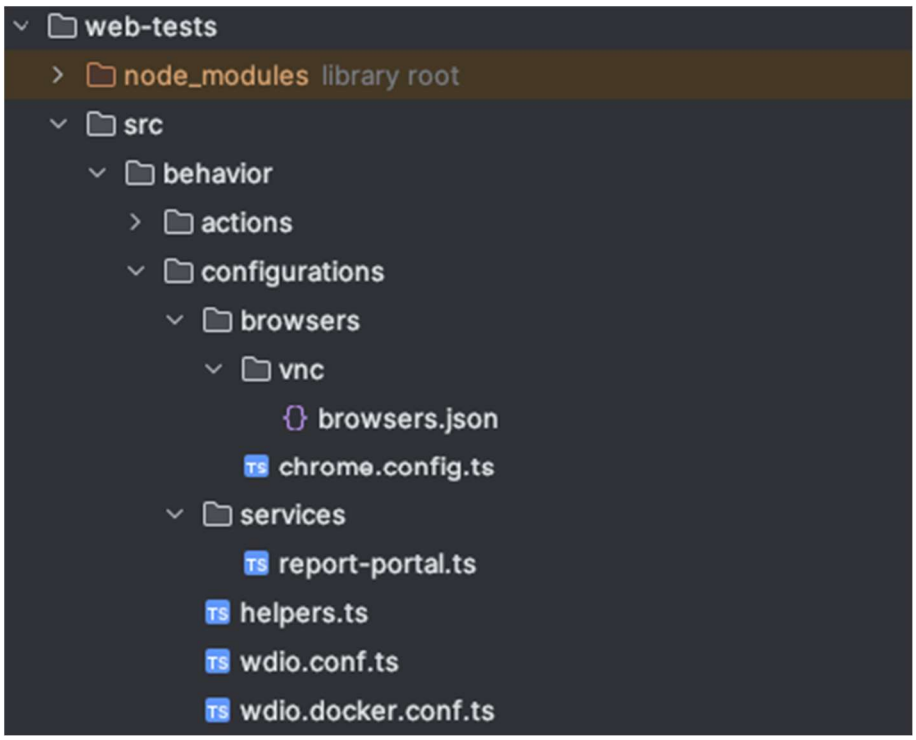

Рисунок 9

Для написания WEB-автотеста необходимо:

- В директории /web-tests/src/features создать директорию для сценария.
- Внутри создать файл с расширением .feature и написать в нем нужный сценарий, максимально переиспользуя уже готовые шаги.
- В директории /web-tests/src/behavior/pageobjects создать (или дополнить существующую) страницу с объектами интерфейса, максимально переиспользуя все, что уже задано ранее. При создании локаторов элементов страниц можно воспользоваться документацией.
- В директории /web-tests/src/behavior/steps создать (или дополнить существующий) файл(ы) с реализацией тех шагов на TypeScript, которые ранее не были реализованы.

3.3. Mobile

Модуль для автотестирования Mobile **mobile-tests** имеет следующие функциональные и конфигурационные директории:

- features (/mobile-tests/src/features). В ней логическим способом по структуре распределяются файлы с расширением .feature, в которых на Gherkin описаны сценарии. Визуально структура отображена на **Рисунок 10**.

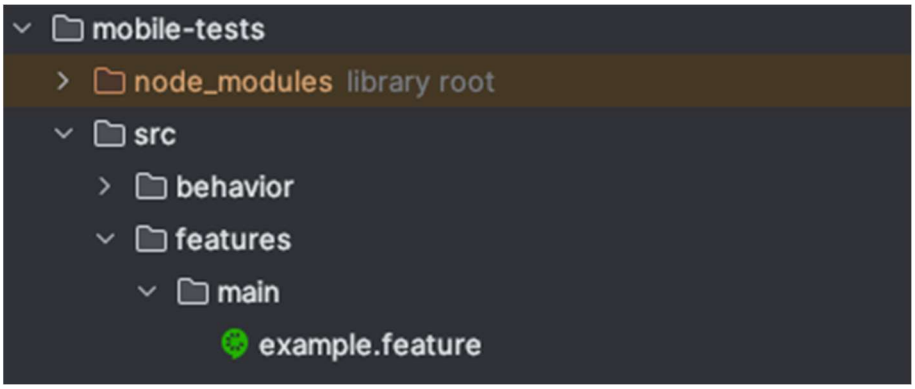

Рисунок 10

- behavior (/mobile-tests/src/behavior). В ней в соответствующих директориях находится реализация, конфигурация и вспомогательные сервисы:
  - actions. Вспомогательные сервисы для шагов Mobile-сценариев (**Рисунок 11**).

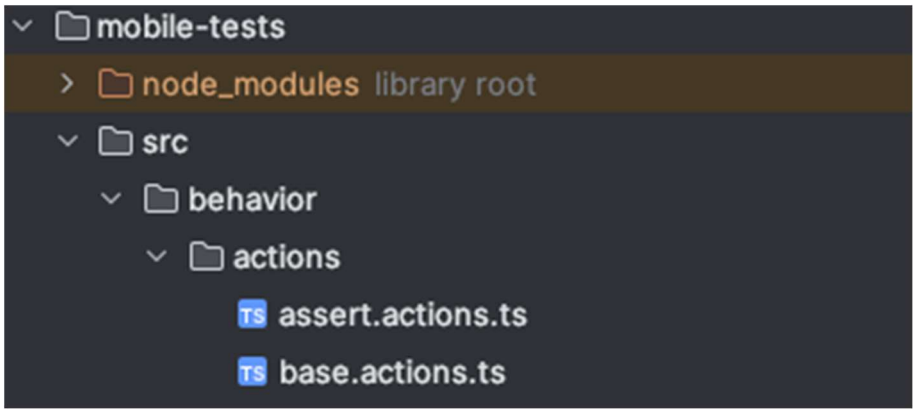

Рисунок 11

  - pageobjects. Описание объектов интерфейса экранов (**Рисунок 12**).

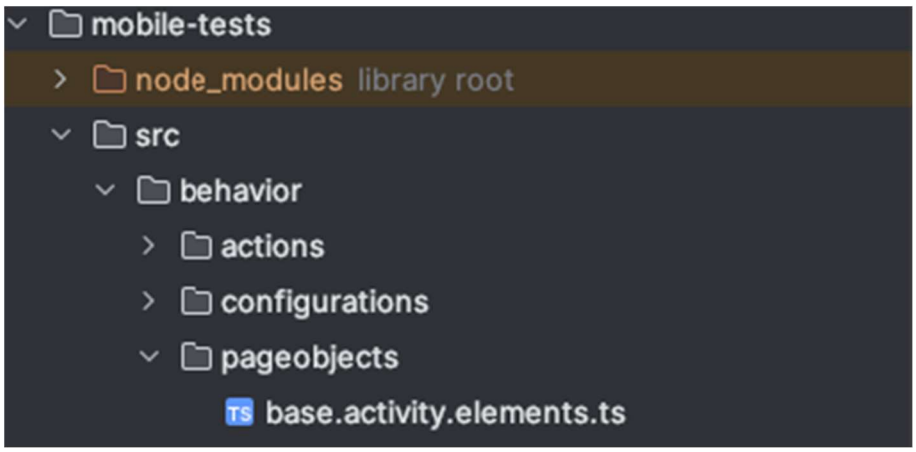

Рисунок 12

- steps. Реализация шагов тестовых сценариев (**Рисунок 13**).

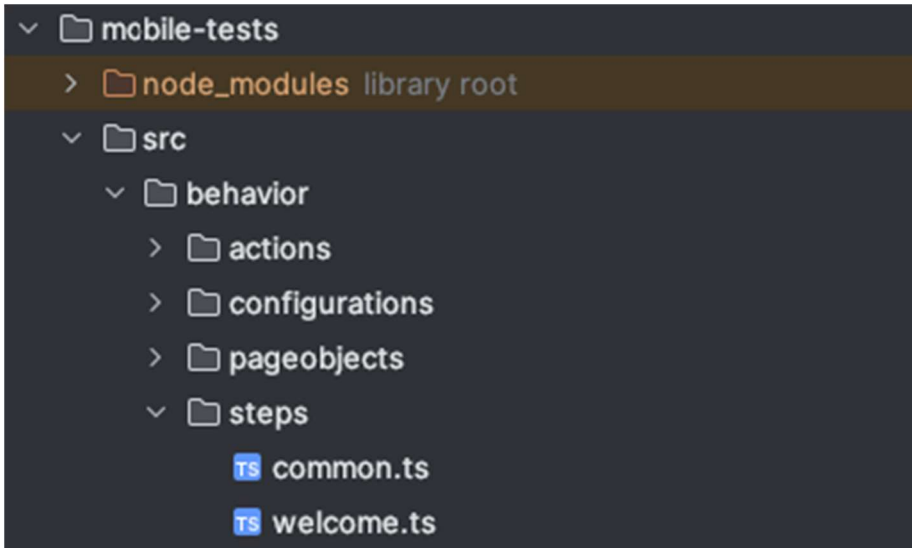

Рисунок 13

- configurations. Конфигурация для отчетности, конфигурация и сервисы для запуска тестов (**Рисунок 14**).

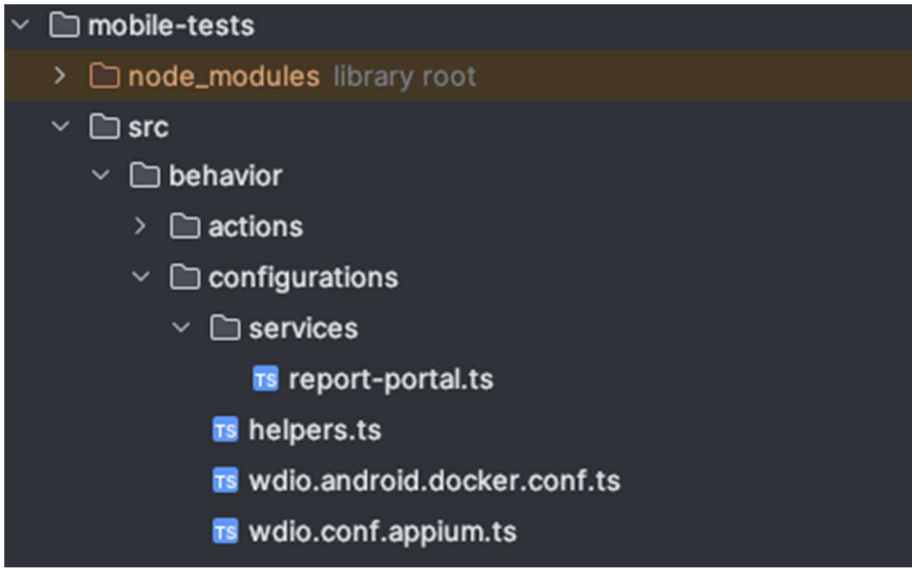

Рисунок 14

Для написания Mobile-автотеста необходимо:

- В директории /mobile-tests/src/features создать директорию для сценария.
- Внутри создать файл с расширением .feature и написать в нем нужный сценарий, максимально переиспользуя уже готовые шаги.
- В директории /mobile-tests/src/behavior/pageobjects создать (или дополнить существующий) экран с объектами интерфейса, максимально переиспользуя все, что уже задано ранее. При создании локаторов элементов экранов можно воспользоваться документацией.

- В директории /mobile-tests/src/behavior/steps создать (или дополнить существующий) файл(ы) с реализацией тех шагов на TypeScrypt, которые ранее не были реализованы.

3.4. Common

Модуль для автотестирования **common** является общим для api-tests, web-tests, mobile-tests и содержит общие шаги реализации (steps) (**Рисунок 15**) и вспомогательные сервисы (actions) (**Рисунок 16**) для этих модулей. Шаги и менеджеры обеспечивают, в частности, интеграционные взаимодействия (Redis, реляционные и нереляционные СУБД, Elasticsearch).

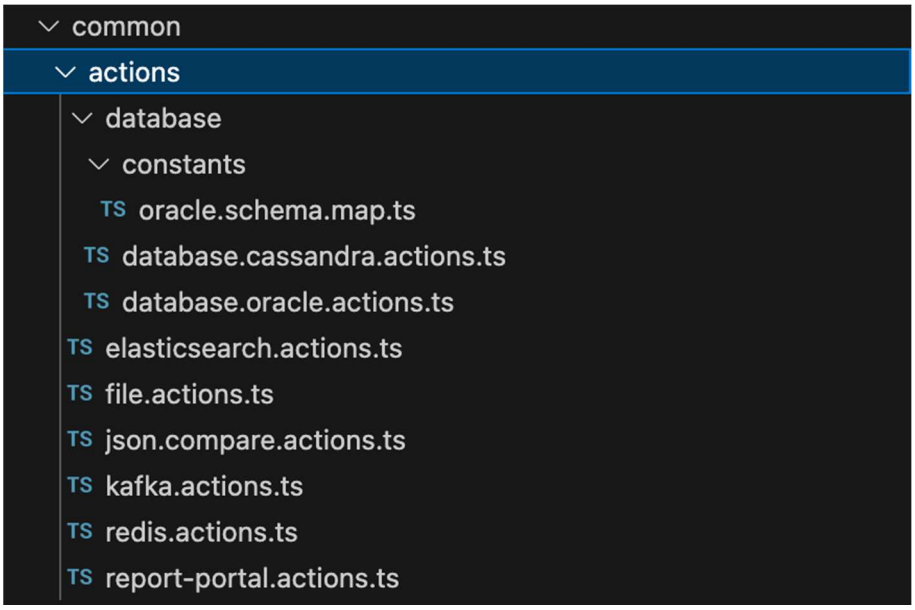

Рисунок 15

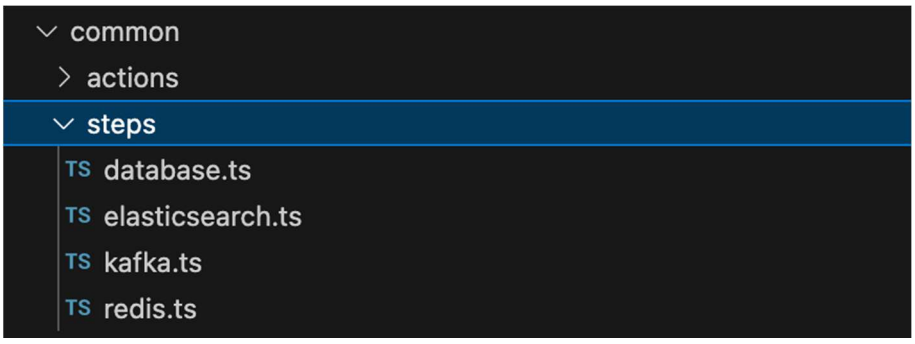

Рисунок 16

4. Контактная информация

В случае возникновения вопросов по настоящей инструкции просим обратиться к следующим специалистам:

Иваньков Иван
Руководитель практики
+7 (903) 141-49-56
iivankov@fil-it.ru

Зубцов Игорь
Руководитель группы тестирования
+7 (910) 283-79-07
izubtsov@fil-it.ru