Инженер по ручному тестированию

01 Кому подойдёт курс

Начинающим без опыта

Сможете получить полное представление об IT и роли QA в нём, а также найти работу в новой профессии

Тем, кто хочет попробовать себя в IT

Освоите востребованную профессию за 4 месяца при нагрузке 12 часов в неделю — без технического бэкграунда

Тем, кто ищет гибкий формат

Подойдёт, если совмещаете учёбу с работой, учёбой или уходом за ребёнком

02 Какие знания и навыки освоите

- Проводить функциональное, UI и регрессионное тестирование веб-приложений
- Анализировать требования, находить «серые зоны» и применять техники тест-дизайна (классы эквивалентности, граничные значения) для создания чек-листов и тест-кейсов
- Оформлять баг-репорты в трекере, классифицировать дефекты по severity/priority, работать в TMS
- Тестировать REST API через Postman, читать документацию Swagger и проверять данные в БД с помощью SQL (SELECT, WHERE, JOIN)
- Использовать DevTools (Elements, Network, Console) для диагностики проблем на фронтенде и фиксации доказательств
- Проводить базовое тестирование мобильных приложений на Android (эмуляторы, Logcat) с учётом особенностей платформы (прерывания, ориентация, жизненный цикл)

03 Как проходит обучение

Курс разбит на спринты по 2 недели. Каждый спринт — это воркбук: вы сразу видите финальный проект, а теорию изучаете ровно в том объёме, который нужен для его выполнения. На каждом спринте — воркшоп с экспертом, где разбирается реальная рабочая задача.

Что вас ждёт на обучении

Все необходимые инструменты и материалы, чтобы начать	Практика, основанная на решении реальных рабочих задач	Обучение от экспертов из Яндекса и других крупных компаний

Инженер по тестированию (базовый)

01 Кому подойдёт курс

Начинающим без опыта

Сможете получить полное представление об IT и роли QA в нём, а также найти работу в новой профессии

Всем, кто хочет развиваться в тестировании

Освоите новую профессию, чтобы сменить сферу, получить повышение на текущей работе либо сменить место работы, оставаясь в профессии

Тем, кто уже знаком с ручным тестированием

Структурируете знания и освоите автоматизацию тестирования на Python — получите сразу две специальности

02 Какие знания и навыки освоите

- Проводить функциональное, UI и регрессионное тестирование веб- и мобильных приложений
- Анализировать требования, находить «серые зоны» и применять техники тест-дизайна (классы эквивалентности, граничные значения) для создания чек-листов и тест-кейсов
- Оформлять баг-репорты в трекере, классифицировать дефекты по severity/priority, работать в TMS
- Тестировать REST API через Postman, читать документацию Swagger и проверять данные в БД с помощью SQL (SELECT, WHERE, JOIN)
- Использовать DevTools (Elements, Network, Console) для диагностики проблем на фронтенде и фиксации доказательств
- Проводить базовое тестирование мобильных приложений на Android (эмуляторы, Logcat) с учётом особенностей платформы
- Писать автотесты для API и веб-интерфейсов на Python с использованием pytest и Playwright
- Работать с Git, создавать ветки, оформлять pull request, разрешать конфликты
- Настраивать CI/CD-пайплайны через GitHub Actions и генерировать отчёты Allure
- Проверять состояние базы данных в автотестах через SQL и psycopg2

03 Как проходит обучение

Курс разбит на спринты по 2 недели. Каждый спринт — это воркбук: вы сразу видите финальный проект, а теорию изучаете ровно в том объёме, который нужен для его выполнения. На каждом спринте — воркшоп с экспертом, где разбирается реальная рабочая задача.

Что вас ждёт на обучении

Все необходимые инструменты и материалы, чтобы начать работать

Практика, основанная на решении реальных рабочих задач

Обучение от экспертов из Яндекса и других крупных компаний

Инженер по тестированию (расширенный)

01 Кому подойдёт курс

Всем, кто хочет развиваться в тестировании

Сможете получить повышение на текущем месте либо сменить место работы, оставаясь в профессии, решая более сложные инфраструктурные задачи

Тем, кто уже прошёл базовый курс или имеет опыт в тестировании

Углубитесь в автоматизацию: освоите архитектуру тестовых фреймворков, Docker, контрактное тестирование и проведение нагрузочного тестирования

02 Какие знания и навыки освоите

- Всё из базового тарифа: ручное тестирование, API-автотесты, UI-автотесты, работа с БД, CI/CD
- Проектировать архитектуру тестовых фреймворков с использованием паттернов PageObject и Service Object
- Контейнеризировать тестовое окружение с помощью Docker и Docker Compose
- Изолировать тесты с помощью моков и стабов (pytest-mock)
- Проводить контрактное тестирование микросервисов (Pact) и тестировать асинхронные системы
- Проводить нагрузочное тестирование и глубоко анализировать производительность системы с помощью Locust

03 Как проходит обучение

Курс разбит на спринты по 2 недели. Каждый спринт — это воркбук: вы сразу видите финальный проект, а теорию изучаете ровно в том объёме, который нужен для его выполнения. На каждом спринте — воркшоп с экспертом, где разбирается реальная рабочая задача.

Что вас ждёт на обучении

Все необходимые инструменты и материалы, чтобы начать работать	Практика, основанная на решении реальных рабочих задач	Обучение от экспертов из Яндекса и других крупных компаний
20 проектов в портфолио + кейсы от Яндекса	Доступ к Мастерской — реальные проекты от заказчиков	Собеседования лучших студентов в компании-партнёры без предварительного отбора

Инженер по ручному тестированию

01	В чём суть профессии	QA-тестировщик — специалист, который проверяет качество IT-продуктов на каждом этапе разработки. Он создаёт тестовые сценарии, находит ошибки и помогает команде выпускать продукты без сбоев. Чем раньше найден баг — тем дешевле его исправить, поэтому тестировщики нужны в каждой IT-компании.
02	В каких сферах работают	Финтех, e-commerce, медтех, геймдев, госсервисы, стартапы — тестировщики нужны везде, где есть цифровой продукт.
03	Какие задачи решают	• Анализируют требования и находят противоречия ещё до начала разработки • Составляют тест-кейсы и чек-листы, проводят ручное тестирование • Тестируют веб- и мобильные приложения, API, базы данных • Оформляют баг-репорты и контролируют исправление дефектов • Готовят отчёты о качестве продукта перед релизом

Зарплатная вилка

81 000 ₽	**147 000 ₽**	**225 000 ₽**
Джуниор инженер по ручному тестированию	Мидл инженер по ручному тестированию	Синьор инженер по ручному тестированию

На курсе — всё, что нужно, чтобы начать карьеру тестировщиком

- Диплом о переподготовке
- Непрерывная практика
- Портфолио из 7 проектов
- Помощь в трудоустройстве

Инженер по тестированию (базовый)

01 В чём суть профессии

Инженер по тестированию — специалист, который обеспечивает качество IT-продуктов на всех этапах разработки. Он не только находит баги вручную, но и автоматизирует повторяющиеся проверки, чтобы команда могла быстрее выпускать надёжные продукты. Это одна из самых востребованных профессий в IT: тестировщики нужны в каждой компании, где есть цифровой продукт.

02 В каких сферах работают

Финтех, e-commerce, медтех, геймдев, госсервисы, стартапы — инженеры по тестированию нужны везде, где есть цифровой продукт и команда разработки.

03 Какие задачи решают

- Анализируют требования и выявляют противоречия ещё до начала разработки
- Составляют тест-кейсы, чек-листы и проводят ручное тестирование веб- и мобильных приложений
- Тестируют API и проверяют данные в базах данных
- Пишут автотесты на Python, которые запускаются автоматически при каждом обновлении продукта
- Настраивают CI/CD-пайплайны и следят за качеством кода через отчёты Allure
- Готовят отчёты о качестве продукта перед релизом

Зарплатная вилка

81 000 ₽
Джуниор инженер по ручному тестированию

147 000 ₽
Мидл инженер по ручному тестированию

225 000 ₽
Синьор инженер по ручному тестированию

На курсе — всё, что нужно, чтобы начать карьеру аналитика

Диплом о переподготовке	Непрерывная практика	Портфолио из 15 проектов	Помощь в трудоустройстве

Собеседования для лучших студентов в компании-партнёры без предварительного отбора

Инженер по тестированию (расширенный)

01	В чём суть профессии	Инженер по тестированию расширенного уровня — специалист, который не просто находит баги и пишет автотесты, но и выстраивает тестовую инфраструктуру для всей команды. Он проектирует фреймворки, настраивает CI/CD, работает с контейнерами, тестирует сложные микросервисные системы и проводит нагрузочное тестирование для оценки производительности под высокой нагрузкой.
02	В каких сферах работают	Финтех, e-commerce, медтех, геймдев, госсервисы, стартапы — тестировщики нужны везде, где есть цифровой продукт.
03	Какие задачи решают	• Проектируют архитектуру тестовых фреймворков и поддерживают их в актуальном состоянии • Настраивают и поддерживают CI/CD-пайплайны для автоматического запуска тестов • Контейнеризируют тестовое окружение с помощью Docker • Проводят нагрузочное тестирование, анализируют метрики производительности и готовят отчёты для разработчиков • Проводят контрактное тестирование микросервисов • Помогают команде выбирать инструменты и подходы к тестированию
04	Карьерные перспективы	Всё, что нужно, чтобы быстрее вырасти до middle и решать сложные задачи.

Зарплатная вилка

81 000 ₽	147 000 ₽	225 000 ₽
Джуниор инженер по ручному тестированию	Мидл инженер по ручному тестированию	Синьор инженер по ручному тестированию

Инженер по тестированию (расширенный)

На курсе — всё, что нужно, чтобы быстрее вырасти до middle и решать сложные задачи

- Диплом о переподготовке
- Непрерывная практика
- 20 проектов в портфолио + кейсы от Яндекса
- Помощь в трудоустройстве
- Доступ к Мастерской — реальные проекты от заказчиков
- Собеседования лучших студентов в компании-партнёры без предварительного отбора

Инженер по тестированию

Сравнение тарифов

Критерий	Инженер по ручному тестированию	Инженер по тестированию (базовый)
Длительность	4 месяца	9 месяцев
Проекты в портфолио	7 + кейсы от Яндекса	15 + кейсы от Яндекса
Воркшопы	8	17
Каникулы	✕	1 неделя
Инструменты		
Ручное тестирование веб и мобильных приложений	✓	✓
Postman, Swagger, DevTools	✓	✓
Яндекс Трекер, Test IT	✓	✓
SQL, DBeaver	✓	✓
Android Studio, Logcat	✓	✓
Python, pytest, requests	✕	✓
Playwright	✕	✓
Allure	✕	✓
Git, GitHub Actions	✕	✓
Карьерные возможности		
Диплом о профессиональной переподготовке	✓	✓
Помощь в трудоустройстве	✓	✓
Вакансии и стажировки от партнёров	✕	✓

Преимущества базового тарифа

01	Две специальности	Получите квалификацию ручного тестировщика и автоматизатора на Python в рамках одной программы
02	Больше проектов	15 проектов в портфолио вместо 7 — больше практики и более сильное резюме
03	Автоматизация	Освоите pytest, Playwright, Allure и GitHub Actions — навыки, которые выделяют вас среди junior-кандидатов
04	Вакансии и стажировки от партнёров	Доступ к эксклюзивным вакансиям и возможность пройти собеседование в компании-партнёры без предварительного отбора

Инженер по тестированию

Сравнение тарифов

Критерий	Инженер по тестированию (базовый)	Инженер по тестированию (расширенный)
Длительность	9 месяцев	12 месяцев
Проекты в портфолио	15 + кейсы от Яндекса	20 + кейсы от Яндекса
Воркшопы	17	21 + 2 сессии ответов на вопросы
Каникулы	1 неделя	2 недели
Инструменты		
Ручное тестирование веб и мобильных приложений	✓	✓
Postman, Swagger, DevTools	✓	✓
SQL, DBeaver, psycopg2	✓	✓
Python, pytest, requests	✓	✓
Playwright	✓	✓
Allure	✓	✓
Git, GitHub Actions	✓	✓
Docker, Docker Compose	✗	✓
pytest-mock (моки и стабы)	✗	✓
Pact (контрактное тестирование)	✗	✓
Locust (нагрузочное тестирование)	✗	✓
Карьерные возможности		
Диплом о профессиональной переподготовке	✓	✓
Помощь в трудоустройстве	✓	✓
Вакансии и стажировки от партнёров	✓	✓
Доступ к Мастерской	✗	✓

Преимущества расширенного тарифа

01	Архитектура фреймворков	Научитесь проектировать масштабируемые тестовые фреймворки с паттернами PageObject и Service Object
02	Docker	Освоите контейнеризацию тестового окружения — навык, который ценится в командах с DevOps-культурой
03	Нагрузочное тестирование	Освоите Locust и научитесь анализировать производительность системы под нагрузкой
04	Углублённое нагрузочное тестирование	Освоите Locust на продвинутом уровне: от написания сценариев до анализа результатов и подготовки отчётов. Сможете проводить комплексное тестирование производительности
05	Мастерская	Получите опыт работы с реальными проектами от коммерческих заказчиков — весомый плюс в резюме

Инженер по ручному тестированию

4 месяца
продолжительность курса

7 проектов
в портфолио

8 воркшопов
с наставником

2 НЕДЕЛИ | 24 ЧАСА

01

Основы тестирования и отчёты об ошибках

2 НЕДЕЛИ | 24 ЧАСА

02

Тест-анализ и тест-дизайн

2 НЕДЕЛИ | 24 ЧАСА

03

Исследовательское и регресс-тестирование

2 НЕДЕЛИ | 24 ЧАСА

04

Тестирование веб-интерфейса

2 НЕДЕЛИ | 24 ЧАСА

05

Тестирование бэкенд: API и базы данных

2 НЕДЕЛИ | 24 ЧАСА

06

Расследование инцидентов: логи и консоль

2 НЕДЕЛИ | 24 ЧАСА

07

Мобильное тестирование

2 НЕДЕЛИ | 24 ЧАСА

08

Итоговая работа: сквозное ручное тестирование

Инженер по тестированию (базовый)

9 месяцев
продолжительность курса

15 проектов
в портфолио

17 воркшопов
с наставником

2 НЕДЕЛИ
01
Основы тестирования и отчёты об ошибках

2 НЕДЕЛИ
02
Тест-анализ и тест-дизайн

2 НЕДЕЛИ
03
Исследовательское и регресс-тестирование

2 НЕДЕЛИ
04
Тестирование веб-интерфейса

2 НЕДЕЛИ
05
Тестирование бэкенд: API и базы данных

2 НЕДЕЛИ
06
Расследование инцидентов: логи и консоль

2 НЕДЕЛИ
07
Мобильное тестирование

2 НЕДЕЛИ
08
Итоговая работа: сквозное ручное тестирование

2 НЕДЕЛИ
09
От ручного тест-кейса к первому автотесту

2 НЕДЕЛИ
10
Тест-дизайн в коде: параметризация и фикстуры

2 НЕДЕЛИ
11
Git: как тестировщики работают в команде

2 НЕДЕЛИ
12
API-тесты: покрываем весь жизненный цикл объекта

2 НЕДЕЛИ
13
Тестируем то, что видит пользователь: первые UI-тесты

2 НЕДЕЛИ
14
Тесты, которым доверяют: отчёты и изоляция

2 НЕДЕЛИ
15
SQL в автотестах: проверяем данные в базе

2 НЕДЕЛИ
16
Тесты запускаются сами: настраиваем CI

2 НЕДЕЛИ
17
Дипломный проект: комплексная автоматизация с БД

Инженер по тестированию (расширенный)

12 месяцев
продолжительность курса

19 проектов
в портфолио

23 воркшопа
с наставником

2 НЕДЕЛИ
01
Основы тестирования и отчёты об ошибках

2 НЕДЕЛИ
02
Тест-анализ и тест-дизайн

2 НЕДЕЛИ
03
Исследовательское и регресс-тестирование

2 НЕДЕЛИ
04
Тестирование веб-интерфейса

2 НЕДЕЛИ
05
Тестирование бэкенд: API и базы данных

2 НЕДЕЛИ
06
Расследование инцидентов: логи и консоль

2 НЕДЕЛИ
07
Мобильное тестирование

2 НЕДЕЛИ
08
Итоговая работа: сквозное ручное тестирование

2 НЕДЕЛИ
09
От ручного тест-кейса к первому автотесту

2 НЕДЕЛИ
10
Тест-дизайн в коде: параметризация и фикстуры

2 НЕДЕЛИ
11
Git: как тестировщики работают в команде

2 НЕДЕЛИ
12
API-тесты: покрываем весь жизненный цикл объекта

2 НЕДЕЛИ
13
Тестируем то, что видит пользователь: первые UI-тесты

2 НЕДЕЛИ
14
Тесты, которым доверяют: отчёты и изоляция

2 НЕДЕЛИ
15
SQL в автотестах: проверяем данные в базе

2 НЕДЕЛИ
16
Тесты запускаются сами: настраиваем CI

Инженер по тестированию (расширенный)

2 НЕДЕЛИ

17

Промежуточный итоговый проект автоматизации

2 НЕДЕЛИ

18

Архитектура тестовых фреймворков

2 НЕДЕЛИ

19

Docker для тестировщика: изолируем окружение

2 НЕДЕЛИ

20

Изоляция через моки: тесты без внешних зависимостей

2 НЕДЕЛИ

21

Тестирование микросервисов: контракты и очереди

2 НЕДЕЛИ

22

Нагрузочное тестирование — часть 1

2 НЕДЕЛИ

23

Нагрузочное тестирование — часть 2 (дипломный проект)

Основы тестирования и отчёты об ошибках 01

2 недели | 24 часа

Познакомитесь с профессией тестировщика, научитесь читать требования и макеты, проводить смоук-тестирование и грамотно документировать найденные дефекты.

Содержание

01.	Анализ входящих артефактов	Работать с текстовыми требованиями и макетами в Figma. Выделять проверяемые ожидания, сопоставлять требования с элементами интерфейса
02.	Тест-исполнение по инструкции	Проводить смоук-тестирование по готовым чек-листам и тест-кейсам. Фиксировать результаты (Passed/Failed), сравнивать ожидаемый и фактический результат
03.	Документирование дефектов	Оформлять баг-репорты по всем правилам: шаги воспроизведения, OP/ФР, окружение, скриншот. Понимать принципы воспроизводимости дефектов
	Проект спринта	Прогон смоук-теста веб-приложения, оформление баг-репортов на найденные дефекты, подготовка сводки результатов

Тест-анализ и тест-дизайн 02

2 недели | 24 часа

Научитесь самостоятельно разбивать требования на объекты тестирования, находить «серые зоны» и создавать качественную тестовую документацию с применением техник тест-дизайна.

Содержание

01.	Декомпозиция требований	Разбивать описание фичи на функциональные объекты тестирования (поля, блоки, действия). Выявлять «серые зоны» в требованиях и формулировать допущения
02.	Применение техник тест-дизайна	Применять классы эквивалентности и граничные значения для формирования тестовых данных. Покрывать максимум сценариев минимальным набором проверок
03.	Создание тестовой документации	Разрабатывать чек-листы и тест-кейсы на основе объектов тестирования. Обеспечивать чёткость и воспроизводимость проверок
	Проект спринта	Самостоятельное создание комплекта тестовой документации для веб-формы — список объектов тестирования, таблица «серые зоны / вопросы / допущения», чек-лист из 15–25 пунктов, 3–7 тест-кейсов

Исследовательское и регресс-тестирование 03

2 недели | 24 часа

Освоите полный цикл регрессионного тестирования и научитесь проводить исследовательские сессии. Поймёте, как классифицировать дефекты и готовить итоговую отчётность.

Содержание

01.	Работа с планом регресс-тестирования	Анализировать готовый план регресса, приоритизировать кейсы в условиях ограниченного времени (таймбокс)
02.	Исполнение тест-рана	Выполнять тест-кейсы и корректно проставлять статусы (Passed, Failed, Blocked). Аргументировать статусы, использовать шаблоны комментариев
03.	Исследовательское тестирование	Проводить сессии исследовательского тестирования по заданному чартеру. Фиксировать находки по шаблону
04.	Классификация дефектов и ретест	Присваивать атрибуты severity и priority баг-репортам. Проводить ретест исправлений и минимальный регресс
05.	Формирование отчётности	Заполнять шаблон отчёта о тестировании: объём прогона, список дефектов, вывод о готовности к релизу
	Проект спринта	Исполнение плана регресс-тестирования, проведение exploratory-сессии, оформление баг-репортов с severity/priority, ретест одного исправления, итоговая сводка

Тестирование веб-интерфейса 04

2 недели | 24 часа

Научитесь проверять интерфейс по макетам, тестировать в разных браузерах и на разных разрешениях, а также использовать DevTools для диагностики проблем на фронтенде.

Содержание

01.	DevTools для диагностики	Использовать вкладки Elements, Network, Console для фиксации доказательств дефектов: визуальные несоответствия, ошибки загрузки, проблемы с сетевыми запросами
02.	Проверка интерфейса по макету	Сверять реализованный интерфейс с макетами в Figma. Фиксировать измеримые несоответствия: отступы, размеры, цвета
03.	Кроссбраузерное и адаптивное тестирование	Проверять корректность отображения и поведения в Chrome и Firefox, а также на разных разрешениях
	Проект спринта	Комплексная проверка интерфейса двух страниц стенда — сверка с макетом, кроссбраузерность, адаптивность, диагностика симптомов через DevTools

Тестирование бэкенд: API и базы данных 05

2 недели | 24 часа

Разберётесь в клиент-серверной архитектуре, научитесь тестировать REST API через Postman и проверять данные в базе данных с помощью SQL-запросов.

Содержание

01.	Основы клиент-серверного взаимодействия	Разбираться в основах HTTP: методы, статус-коды. Анализировать запросы и ответы через вкладку Network в DevTools
02.	Исследование API с помощью Postman	Работать с документацией API (Swagger). Создавать и отправлять запросы (GET, POST) в Postman, анализировать ответы в формате JSON
03.	Тест-дизайн для API	Применять техники тест-дизайна (КЭ, ГЗ) к параметрам API. Создавать сценарии с валидными и невалидными данными
04.	Проверка данных в БД с помощью SQL	Писать базовые SQL-запросы (SELECT, WHERE, JOIN) в DBeaver для проверки данных, изменённых через API
05.	Сквозная проверка API и БД	Проводить сквозную проверку: выполнить действие через API → проверить корректность ответа → проверить изменения в базе данных
	Практикум спринта	Сквозное тестирование API и БД — 3 запроса через Postman (GET, POST, PUT/DELETE), проверка ответов, 2 SQL-запроса для проверки данных, сквозная связка «запрос → ответ → БД»

Расследование инцидентов: логи и консоль 06

2 недели | 24 часа

Научитесь работать в командной строке, находить и анализировать серверные логи, фиксировать доказательства из логов в баг-репорте.

Содержание

01.	Основы работы в командной строке (CLI)	Подключаться к серверу по SSH. Использовать базовые команды навигации (ls, cd, cat) для поиска файлов логов
02.	Поиск и анализ логов	Понимать структуру логов. Фильтровать логи по времени, уровню ошибки (ERROR) и ключевым словам с помощью grep и tail
03.	Фиксация доказательств из логов	Сохранять релевантные фрагменты логов. Грамотно оформлять найденные доказательства в баг-репорте
	Проект спринта	Расследование инцидента на учебном сервере — подключение по SSH, поиск лога за указанный период, фильтрация ошибок (grep ERROR), оформление баг-репорта с приложенным фрагментом лога

Мобильное тестирование 07

2 недели | 24 часа

Освоите специфику тестирования мобильных приложений на Android: настроите эмулятор, проведёте тестирование по чек-листу и научитесь снимать логи Logcat для диагностики сбоев.

Содержание

01.	Особенности мобильного контекста	Разбираться в типах мобильных приложений (native, web, hybrid). Учитывать специфику тестирования: установка, обновление, прерывания, ориентация экрана
02.	Организация тестового окружения	Настраивать эмулятор Android в Android Studio. Устанавливать тестовое приложение на эмулятор
03.	Ключевые сценарии мобильного тестирования	Тестировать мобильное приложение по чек-листу: установка, основные функции, прерывания (звонок, уведомление), смена ориентации
04.	Диагностика сбоев через Logcat	Снимать и базово анализировать логи мобильного приложения через Logcat для диагностики сбоев
	Проект спринта	Тестирование мобильного приложения (Android) по готовому чек-листу на эмуляторе, фиксация дефектов с учётом контекста устройства, снятие логов Logcat при сбое

Итоговая работа: сквозное ручное тестирование 08

2 недели | 24 часа

Финальный проект курса. Самостоятельно протестируете цифровой продукт — веб-приложение, мобильную версию и API — и подготовите полный пакет тестовой документации и итоговый отчёт.

Содержание

01.	Самостоятельная работа	Провести анализ требований, составить чек-лист и тест-кейсы, выполнить тестирование веб-, мобильной версии и API, завести баг-репорты, подготовить итоговый отчёт для команды.
	Итоговый проект	Функциональное тестирование цифрового продукта с подготовкой полного пакета тестовой документации и отчёта

Далее — спринты только базового и расширенного тарифов

Вход в автоматизацию. Первый тест 09

2 недели | 24 часа

Сделаете первый шаг в автоматизацию: разберётесь, какие тесты стоит автоматизировать, освоите основы Python и напишете первый API-автотест с помощью pytest и requests.

Содержание

01.	Первый автотест: от ручного кейса к коду	Понимать виды тестов в пирамиде (unit, integration, e2e) и выбирать, что стоит автоматизировать. Читать ручной тест-кейс и переносить шаги в код: действия — в requests, ОР — в assert
02.	Python: синтаксис и типы	Работать с базовыми типами (int, str, bool, None), f-строками для формирования URL и сообщений, словарями и списками для работы с JSON
03.	Python: управление потоком и окружение	Создавать виртуальное окружение (venv), устанавливать библиотеки, организовывать структуру проекта. Хранить чувствительные данные в .env через python-dotenv
04.	Git: первое знакомство	Клонировать репозиторий, добавлять файлы, делать commit и push, создавать .gitignore и минимальный README с инструкцией запуска
	Проект спринта	Написать и запустить первый API-автотест с использованием pytest и requests. Закоммитить код в Git-репозиторий с README

Тест-дизайн в коде: параметризация и фикстуры 10

2 недели | 24 часа

Научитесь организовывать тесты: выносить код в функции и модули, читать тестовые данные из файлов, параметризовать тесты и использовать фикстуры.

Содержание

01.	Функции и модули	Выносить повторяющийся код в функции, разбивать проект на несколько файлов и импортировать функции между модулями
02.	Работа с данными и ошибками	Читать тестовые данные из JSON-файла через with open и json.load. Обрабатывать исключения requests (ConnectionError, Timeout, HTTPError) через try/except. Писать негативный тест на статус 404 через assert
03.	Pytest: параметризация и фикстуры (база)	Применять @pytest.mark.parametrize для прогона тестов на нескольких наборах данных. Создавать фикстуры с yield для подготовки и очистки данных. Использовать маркеры skip и xfail
04.	Организация тестов	Организовывать тесты в структурированный проект с conftest.py для переиспользования фикстур. Запускать конкретные тесты с флагами -v и -k, читать traceback pytest. Писать assert с информативным сообщением через f-строку
	Проект спринта	Рефакторинг тестов — вынести код в функции, добавить параметризацию и фикстуры, читать тестовые данные из JSON-файла

ООП и Git: как тестировщики работают в команде 11

2 недели | 24 часа

Освоите основы ООП и примените их на практике — вынесете логику работы с API в класс-хелпер. Научитесь работать с ветками Git и оформлять pull request.

Содержание

01.	ООП в Python	Создавать классы с init и self, использовать наследование и super(). Применять @dataclass для хранения данных
02.	Рефакторинг	Выносить повторяющуюся логику в класс-хелпер (например, BaseApiClient → OrdersApiClient). Проверять, что поведение тестов после рефакторинга не изменилось. Оценивать качество кода до и после
03.	Git для командной работы	Создавать ветки, выполнять слияние, оформлять pull request и проходить code review. Разрешать конфликты слияния
	Проект спринта	Рефакторинг проекта с выделением API-хелпера в класс, зафиксировать изменения в ветке и оформить PR

API-тесты: покрываем весь жизненный цикл объекта 12

2 недели | 24 часа

Соберёте полный набор CRUD-тестов с авторизацией через токен, освоите продвинутые фикстуры и научитесь валидировать схемы ответов.

Содержание

01.	Продвинутые фикстуры	Создавать фикстуры с нужной областью видимости (function, module, session). Организовывать фикстуры в conftest на разных уровнях проекта. Реализовывать фикстуру session scope для получения и хранения токена
02.	API: CRUD и авторизация	Реализовывать тесты на создание, чтение, обновление и удаление сущности. Понимать разницу PUT (полная замена) и PATCH (частичное обновление). Организовывать проект по слоям: клиенты, хелперы, фикстуры, тесты
03.	Валидация и качество	Описывать модели данных через pydantic v2 и валидировать ответы API через model_validate(). Писать тесты по принципу AAA с понятными именами и информативными сообщениями в assert
	Проект спринта	CRUD-набор тестов для API-сущности с авторизацией через токен, проверкой схемы ответа через pydantic и структурой проекта по слоям

Тестируем то, что видит пользователь: первые UI-тесты 13

2 недели | 24 часа

Освоите автоматизацию тестирования веб-интерфейса с Playwright — найдёте элементы на странице, реализуете сценарии взаимодействия и научитесь бороться с нестабильностью тестов.

Содержание

01.	Основы Playwright	Устанавливать Playwright, понимать иерархию browser/context/page. Выбирать устойчивые локаторы (get_by_role, get_by_text, CSS, XPath) и оценивать их надёжность
02.	Взаимодействие и проверки	Реализовывать сценарии с формами (click, fill), использовать expect для проверки состояний. Применять автоожидания Playwright, избегать time.sleep. Настраивать скриншоты при падении теста
03.	Организация UI-тестов	Выносить инициализацию браузера и страницы в фикстуры. Организовывать UI-тесты в структурированный проект с conftest.py
	Проект спринта	Написать стабильные UI-тесты для критичного пути веб-приложения с автоожиданиями и скриншотами при падении

Тесты, которым доверяют: отчёты и изоляция 14

2 недели | 24 часа

Подготовите тест-сьют к командной работе: добавите детализированную отчётность через Allure, обеспечите изоляцию тестов и реализуете негативные сценарии с проверкой исключений.

Содержание

01.	Allure	Подключать allure-pytest, разбивать тест на шаги через @allure.step, прикреплять JSON-ответы и скриншоты. Расставлять метки severity и story для группировки тестов
02.	Изоляция тестов	Создавать фикстуры, которые готовят данные через API и удаляют их после теста. Проверять независимость тестов при запуске в случайном порядке. Сравнивать стратегии управления данными (API, БД, транзакции)
03.	Негативные сценарии и pydantic	Использовать pytest.raises для проверки исключений (ConnectionError, Timeout, HTTPError). Валидировать схему ответа через pydantic v2. Воспроизводить упавший запрос через curl или Postman по данным из Allure
	Проект спринта	Добавить Allure к тестам, обеспечить изоляцию данных через фикстуры, реализовать негативные тесты с pytest.raises и валидацию схемы через pydantic

SQL в автотестах: проверяем данные в базе 15

2 недели | 24 часа

Научитесь подключаться к базе данных из автотестов и добавлять проверку состояния данных в БД после выполнения API-запросов.

Содержание

01.	Подключение к БД из Python	Подключаться к PostgreSQL через psycopg2 с помощью контекстного менеджера with. Выполнять SELECT через fetchone/fetchall, писать безопасные параметризованные запросы. Хранить параметры подключения в .env
02.	Сквозная проверка данных	Добавлять DB-check в API-тест: после вызова API проверять через SELECT, что запись появилась в БД. Связывать данные через JOIN и WHERE. Использовать SQL для диагностики падений тестов
03.	Управление данными через БД	Создавать фикстуры для подготовки и очистки данных через БД. Сравнивать стратегии изоляции (API, БД, транзакции) и обосновывать выбор для конкретного сценария
	Проект спринта	Добавить DB-check в один из существующих API-тестов, реализовать фикстуру для очистки данных через БД

Тесты запускаются сами: настраиваем CI 16

2 недели | 24 часа

Настроите автоматический запуск тестов в GitHub Actions при каждом обновлении кода и организуете публикацию Allure-отчётов.

Содержание

01.	Основы CI и pipeline	Понимать что такое CI, pipeline, job, step. Писать GitHub Actions workflow в YAML. Добавлять кэширование зависимостей через setup-python cache и сохранять Allure-отчёт как артефакт
02.	Данные и безопасность в CI	Добавлять секреты в GitHub и безопасно передавать токены и параметры подключения к БД в workflow
03.	Диагностика и качество pipeline	Читать логи CI и находить причины падений. Проверять стабильность pipeline при повторном запуске
	Проект спринта	Настроить GitHub Actions workflow, который при пуше в main запускает тесты, сохраняет Allure-отчёт как артефакт и публикует его на GitHub Pages

Дипломный проект: комплексная автоматизация 17

2 недели | 24 часа

Для базового тарифа

Финальный проект курса. Самостоятельно разработаете полноценный набор автотестов для реального учебного продукта, применив все освоенные инструменты.

Для расширенного тарифа

Промежуточный итоговый проект автоматизации. Самостоятельно разработаете полноценный набор автотестов, объединив все инструменты базовой части курса.

Архитектура тестовых фреймворков 18

2 недели | 24 часа

Научитесь проектировать масштабируемые тестовые фреймворки: освоите паттерны PageObject и Service Object, научитесь правильно организовывать фикстуры и писать API-клиенты.

Содержание

01.	Слоистая архитектура тестового проекта	Понимать принципы слоистой архитектуры: разделение на API-клиенты, модели данных, фикстуры и тесты. Применять паттерны PageObject для UI и Service Object для API
02.	Модели данных и фикстуры	Описывать модели данных через pydantic. Организовывать фикстуры в conftest.py для переиспользования на разных уровнях проекта
03.	Рефакторинг существующего проекта	Проводить рефакторинг существующего проекта: выделять API-клиент, создавать Page Object, перерабатывать фикстуры. Проверять, что поведение тестов не изменилось
	Проект спринта	Рефакторинг проекта из прошлых спринтов — выделить API-клиент, создать Page Object для UI-тестов, вынести модели данных (pydantic), переработать фикстуры в conftest.py

Docker для тестировщика: изолируем окружение 19

2 недели | 24 часа

Освоите контейнеризацию тестового окружения с помощью Docker — научитесь запускать тесты в изолированных контейнерах и интегрировать Docker в CI.

Содержание

01.	Основы Docker	Понимать ключевые концепции Docker: образы, контейнеры, Dockerfile. Создавать образ для тестового проекта
02.	Docker Compose	Описывать многоконтейнерное окружение в docker-compose.yml. Поднимать тестовую БД и запускать тесты внутри контейнера
03.	Интеграция Docker с CI	Встраивать Docker в GitHub Actions pipeline. Обеспечивать стабильный запуск тестов в контейнеризированном окружении
	Проект спринта	Разработать Dockerfile и docker-compose.yml для тестового проекта, поднять тестовую БД в контейнере и обеспечить успешный запуск тестов

Изоляция через моки: тесты без внешних зависимостей 20

2 недели | 24 часа

Научитесь изолировать тесты от внешних зависимостей с помощью моков и стабов — эмулировать ответы API, ошибки и таймауты без реальных обращений к сервисам.

Содержание

01.	Моки и стабы	Использовать pytest-mock (unittest.mock) для подмены возвращаемых значений. Проверять вызовы через call_count и assert_called_with. Эмулировать исключения (ошибка 500, таймаут)
02.	Мокирование API-клиента	Мокировать функции-обёртки над requests вместо реальных вызовов. Проверять, что при ошибке функция возвращает None или выбрасывает исключение
	Проект спринта	Для одного API-эндпоинта написать тесты с моками — эмулировать успешный ответ, ошибку 500 и таймаут. Проверить поведение функции при каждом сценарии

Тестирование микросервисов: контракты и очереди 21

2 недели | 24 часа

Освоите специализированные подходы к тестированию сложных систем: контрактное тестирование микросервисов с Pact и тестирование асинхронных систем с очередями сообщений.

Содержание

01.	Контрактное тестирование с Pact	Понимать принципы контрактного тестирования. Писать consumer-тесты с Pact и верифицировать провайдера
02.	Тестирование асинхронных систем	Понимать основы очередей сообщений (RabbitMQ/Kafka). Проверять отправку сообщения в очередь и его обработку с помощью testcontainers или моков
	Проект спринта	Написать тест для проверки отправки и обработки сообщения в очереди RabbitMQ

Нагрузочное тестирование — часть 1 22

2 недели | 24 часа

Освоите основы нагрузочного тестирования: научитесь писать сценарии в Locust, собирать ключевые метрики производительности и анализировать базовые результаты.

Содержание

01.	Виды нагрузочного тестирования	Разбираться в отличиях нагрузочного, стресс-тестирования и тестирования стабильности. Понимать цели каждого подхода
02.	Locust: архитектура и написание сценариев	Устанавливать Locust, писать сценарии нагрузки с использованием Tasks и TaskSets. Моделировать поведение пользователей
03.	Ключевые метрики производительности	Собирать и анализировать метрики: RPS (запросов в секунду), latency (задержка), error rate (частота ошибок)
	Проект спринта	Написать сценарий Locust для API эндпоинта, провести тест с постепенно возрастающей нагрузкой, собрать базовые метрики

Нагрузочное тестирование — часть 2 (дипломный проект) 23

2 недели | 24 часа

Финальный проект курса. Проведёте комплексное нагрузочное тестирование, проанализируете результаты, найдёте узкие места и подготовите профессиональный отчёт.

Содержание

01.	Продвинутые сценарии нагрузки	Писать сложные сценарии с разными профилями пользователей, авторизацией, обработкой динамических данных
02.	Анализ результатов и поиск узких мест	Анализировать графики производительности, находить точки насыщения системы, определять узкие места
03.	Подготовка отчёта и рекомендаций	Составлять профессиональный отчёт о нагрузочном тестировании с выводами, графиками и рекомендациями по оптимизации
	Итоговый проект	Провести нагрузочное тестирование учебного сервиса, включая написание сценариев, запуск тестов с разными профилями нагрузки, анализ результатов и подготовку итогового отчёта с рекомендациями