Утверждено
Приказом Генерального директора
№ 16-О от 10.08.2026
Генеральный директор
ООО «НТЦ ПРОТЕЙ»

Апостолова Н.А.

ПРОТЕЙ

Программа курса
«QA-инженер»

Санкт-Петербург
2026 г.

Содержание

Длительность курса

Программа обучения рассчитана на 52 часов. Количество часов, отводимых на изучение теоретической программы – 20 часов, отработка практических навыков – 32 часов.

Цель курса

Подготовка начинающих специалистов и студентов технических направлений к профессиональной деятельности в качестве инженера по тестированию программного обеспечения (QA Engineer) с применением технологий и методологий, принятых в качестве технологического стека ГК ПРОТЕЙ.

Категория обучающихся

Студенты и начинающие специалисты IT- и телеком-направлений, желающие овладеть знаниями и навыками тестирования программного обеспечения, включая проведение ручного функционального тестирования, работу с требованиями и баг-трекинговыми системами, а также получение базовых навыков работы с инструментами анализа сетевого трафика и API.

Уровень обучающихся

- имеют базовые знания в области информационных технологий;
- имеют навыки программирования на Python/Java;
- имеют практические навыки работы на средствах вычислительной техники.

Планируемые результаты обучения

После прохождения курса обучающиеся
будут знать:

- роль тестировщика, принципы тестирования, жизненные циклы SDLC и STLC;
- виды и уровни тестирования;
- техники тест-дизайна: классы эквивалентности, граничные значения, таблицы решений, переходы состояний и попарное тестирование;
- основы тестовой стратегии, оценки покрытия и использования метрик;
- основные архитектурные подходы и их влияние на тестирование;
- основы сетевого взаимодействия, протоколов HTTP/HTTPS, DNS и TLS;
- принципы работы REST API, форматы JSON, XML и Protobuf;
- основы Linux, баз данных, Docker, Git и CI/CD;
- принципы построения API- и UI-автотестов;
- возможности, ограничения и риски применения AI-инструментов в тестировании

будут уметь

- анализировать требования, техническую документацию и архитектуру системы;

- составлять тест-планы, чек-листы, тест-кейсы;
- применять техники тест-дизайна и определять пробелы тестового покрытия;
- проводить исследовательское, функциональное и интеграционное тестирование;
- оформлять понятные и воспроизводимые баг-репорты с указанием критичности и приоритета;
- анализировать DOM, консоль, сетевые запросы и ответы через Browser DevTools;
- использовать Linux-инструменты для анализа логов, процессов, ресурсов и сетевых проблем;
- выполнять SQL-запросы для подготовки и проверки тестовых данных;
- разрабатывать API- и UI-автотесты на Python или Java;
- запускать тестовое окружение в Docker и автотесты в CI/CD;
- анализировать результаты тестирования и формировать заключение о готовности продукта к релизу;
- использовать AI-инструменты для решения QA-задач и проверять корректность полученного результата.

Материально-техническое обеспечение

Проектор с воспроизведением на интерактивную панель, маркерная доска, учебная мебель (стулья, столы для слушателей в достаточном количестве), стол и стул для преподавателя, стеллаж для хранения документов, необходимое количество ноутбуков для работы обучающихся, а также ноутбук для преподавателя.

Учебный план

№ п/п	Наименование разделов и тем	Всего, часов	Теория	Практика	Форма проведения занятия
1.	Тема 1. Основы тестирования	5	2	3	Лекция, Практическое задание
2.	Тема 2. Стратегия и метрики тестирования	5	2	3	Лекция, Практическое задание
3.	Тема 3. Архитектура программного обеспечения	5	2	3	Лекция, Практическое задание
4.	Тема 4. Сети, Linux	5	2	3	Лекция, Практическое задание
5.	Тема 5. Автоматизация тестирования API	6	2	4	Лекция, Практическое задание
6.	Тема 6. Автоматизация тестирования UI	6	2	4	Лекция, Практическое задание
7.	Тема 7. Базы данных для тестировщика	6	2	4	Лекции, Практическое задание
8.	Тема 8. Docker, CI/CD	6	2	4	Лекция, Практическое задание
9.	AI-инструменты в работе тестировщика	6	2	4	Лекция, Практическое задание
10.	Итоговая практика и защита проекта	2	2	-	Лекция, Практическое задание (выпускная работа)
Итого:		52	20	32	

Рабочая программа

Тема 1. Основы тестирования

- Тестировщик и его роль (QA, QC, Testing).
- Принципы тестирования.
- Жизненный цикл ПО - SDLC, STLC.
- Уровни тестирования, пирамидка (модульное, интеграционное, системное, приёмочное).
- Виды тестирования: функциональное, нефункциональное (нагрузка, безопасность, удобство использования).
- Артефакты тестирования: тест-кейс, чек-лист, баг-репорт
- Статическое vs динамическое тестирование.
- Техники тест-дизайна: классы эквивалентности, анализ граничных значений.
- Продвинутые техники: таблицы решений, переходы состояний, попарное покрытие.
- Исследовательское тестирование.

Тема 2. Стратегия и метрики тестирования

- Модели разработки (Waterfall, Agile, Scrum, Kanban).
- Стратегия тестирования: risk-based, requirements-based, model-based.
- Тестовое покрытие: требования, функциональность, код; анализ пробелов покрытия.
- Метрики процесса: процент автоматизированных тестов, время выполнения, pass rate.
- Метрики продукта: плотность дефектов, классификация дефектов и уровень обнаружения.
- Принятие решения о релизе на основе метрик.
- Quality gates.

Тема 3. Архитектура программного обеспечения

- Виды архитектур: монолитная, многослойная, многоуровневая, клиент-серверная, сервис-ориентированная (SOA), микросервисная и т.д.
- Основные компоненты систем: клиенты, backend, gw, db, cache, kv, broker, s3, балансировщики и прокси.
- Взаимодействие компонентов: синхронные и асинхронные вызовы, API-контракты, очереди и события.
- Влияние архитектуры на стратегию, уровни и объём тестирования.
- Тестирование распределённых систем: тайм-ауты, повторные запросы, дублирование сообщений, нарушение порядка, частичные отказы, идемпотентность.
- Отказоустойчивость и масштабирование: retry, circuit breaker, репликация, балансировка нагрузки и их проверка.
- Тестовые окружения и изоляция зависимостей: mock, stub, fake, эмуляторы, service virtualization.

- Наблюдаемость и диагностика дефектов: логи, метрики, трассировка, correlation ID, health checks.
- Чтение и визуализация архитектуры: component, deployment и sequence diagrams, PlantUML.

Тема 4. Сети, Linux

- Основы сетевого взаимодействия: модель OSI и стек TCP/IP, IP-адреса, порты, TCP и UDP.
- Основные протоколы: HTTP/HTTPS, DNS, TLS; DHCP.
- HTTP: методы, коды состояния, заголовки, тело запроса и ответа, cookies, авторизация и редиректы.
- Диагностика сети: curl, ping, traceroute, dig, nslookup, nc, ss, lsof.
- Анализ трафика: Browser DevTools, Charles Proxy / Proxyman, Wireshark, tcpdump.
- Основы Linux: дистрибутивы, файловая система, переменные окружения, процессы и сервисы.
- Пакеты и пакетные менеджеры: DEB/RPM/APK, apt, dnf/yum, apk, репозитории, версии и зависимости.
- Управление системой: ps, top, kill, systemctl, права доступа chmod и chown.
- Диагностика приложения: логи (tail, less, journalctl) и ресурсы (htop, free, df, du, iotop).
- Типовые проблемы окружения: отсутствующая зависимость, несовместимая версия, неправильная конфигурация, занятый порт, недостаточные права.

Тема 5. Автоматизация тестирования API

- Виды API: REST/HTTP, SOAP, RPC/gRPC, GraphQL; форматы данных JSON, XML, Protobuf.
- Структура API запроса: URL, параметры, заголовки, cookies и тело.
- Контракты API: OpenAPI/Swagger, схемы данных, обязательные поля, типы, ограничения и обратная совместимость.
- Тест-дизайн API: позитивные и негативные проверки, валидация данных, обработка ошибок.
- Проверка поведения API: CRUD, идемпотентность, пагинация, фильтрация, сортировка, повторные и параллельные запросы.
- Аутентификация и авторизация: Basic Auth, API Key, Bearer Token, JWT, OAuth 2.0, роли и проверка прав доступа.
- Ручное тестирование: Postman коллекции, окружения, переменные, скрипты и цепочки запросов; curl.
- Структура автотестов API: клиент API, подготовка данных, параметризация, ассерты, логирование и очистка данных.
- Реализация на Python: pytest + requests/httpx; на Java: JUnit/TestNG + REST Assured.

- Проверки в автотестах: статус, заголовки, тело, JSON Schema, время ответа и побочные изменения в системе.
- Эмуляция зависимостей: WireMock, Mockoon; настройка ответов, ошибок, задержек и проверка полученных запросов

Тема 6. Автоматизация тестирования UI

- Инструменты UI-автоматизации: Python: Playwright + pytest; Java: Selenide + JUnit/TestNG; Selenium WebDriver.
- DOM и поиск элементов: CSS, XPath, текстовые, role- и test-id-локаторы.
- Синхронизация с интерфейсом: автоматические и явные ожидания, тайм-ауты, причины нестабильных тестов.
- Работа с Web UI: формы, таблицы, файлы, iframe, shadow DOM, диалоговые окна и вкладки.
- Архитектура UI-тестов: Page Object, Component Object, разделение тестовой логики, данных и проверок.
- Подготовка и изоляция тестов: авторизация, cookies, состояние браузера, создание данных через API.
- Перехват и эмуляция сетевых запросов, тестирование ошибок и задержек backend.
- Диагностика тестов: скриншоты, видео, trace, логи браузера и сетевые запросы.

Тема 7. Базы данных для тестировщиков

- Роль базы данных в тестировании: источник состояния системы, проверка результатов и локализация дефектов.
- Структура реляционных БД: таблицы, связи, первичные и внешние ключи, ограничения, NULL.
- SQL для проверки данных: SELECT, WHERE, JOIN, GROUP BY, подзапросы и агрегатные функции.
- Проверка изменений после UI- и API-операций: созданные записи, связанные сущности, статусы, история изменений.
- Подготовка и очистка тестовых данных: INSERT, UPDATE, DELETE, фикстуры и независимость тестов.
- Транзакции и целостность данных: COMMIT, ROLLBACK, частично выполненные операции, блокировки и параллельные изменения.
- Типовые дефекты данных: дубли, потеря записей, неверные связи, рассинхронизация, нарушение ограничений.
- Миграции и совместимость схемы: добавление полей, изменение типов, значения по умолчанию и сохранность старых данных.
- Индексы и планы выполнения запросов: поиск причин медленных операций.
- Инструменты: DBeaver, DataGrip, pgAdmin, psql.

Тема 8. Docker, CI/CD

- Основы Git для тестировщика: commit, branch, merge/rebase, pull/merge request, разрешение конфликтов и code review.
- CI/CD: конвейер сборки и тестирования (Jenkins, GitLab CI)
- Запуск автотестов в пайплайне: параллелизация, артефакты (report, скриншоты, trace)
- Интеграция Docker в CI/CD: сборка образа с тестами, запуск контейнера в пайплайне
- Параметризованные запуски, ночные прогоны
- Основы Docker: образ, контейнер, registry, Dockerfile, volume, network, port mapping.
- Работа с контейнерами: docker ps, run, exec, logs, inspect, stop, rm.
- Docker Compose: запуск системы из нескольких сервисов, зависимости, переменные окружения и конфигурация.
- Диагностика проблем в CI: упавший контейнер, недоступный сервис, неверная конфигурация, ошибка образа или самого теста.

Тема 9. AI-инструменты в работе тестировщика

- Основы LLM: модели, контекст, токены, промпты и ограничения.
- Разница между чатом и локальным AI-агентом с доступом к проекту и терминалу.
- Основные сущности: провайдеры, модели, инструменты, агенты, скиллы и workflow.
- Применение ИИ в QA: анализ требований, генерация проверок, написание тестов, анализ логов и ошибок.
- Основы OpenCode: установка, подключение модели, конфигурация проекта и разрешения.
- Работа со скиллами: назначение, структура, входные данные и формат результата.
- Безопасность: секреты, персональные данные, контроль команд и проверка изменений.
- Проверка результата ИИ: просмотр diff, запуск тестов, линтеров и ручная валидация.
- Типовые ошибки: выдуманные библиотеки и методы, лишние изменения, нестабильные тесты, слабые проверки и подгонка теста под текущую реализацию.

Тема 10. Итоговая практика и защита проекта

- Обзор архитектуры проектов автотестов на Python и Java.
- Постановка цели и объяснение задач выпускной работы.

Тематика практических заданий

Задание 1. Основы тестирования, тест-дизайн

- Составление чек-листа и тест-кейсов для формы авторизации (логин/пароль) с использованием техник: классы эквивалентности, граничные значения
- Составление таблицы решений для выданной формы
- Проведение сессии исследовательского тестирования реального веб-приложения с заполнением списка найденных дефектов.
- Оформление баг-репортов для найденных дефектов

Задание 2. Стратегия и метрики тестирования

- Разработка тест-плана для небольшого проекта (например, интернет-магазин) с разбивкой по уровням и видам тестирования.
- Построить матрицу трассировки требований и определить пробелы покрытия.
- Рассчитать метрики по предоставленным данным и принять решение о релизе на основании заданных Quality Gates.

Задание 3. Архитектура программного обеспечения

- Проанализировать архитектуру учебного приложения: определить компоненты, зависимости и способы взаимодействия.
- Построить component диаграмму в PlantUML.
- Определить точки интеграционного тестирования, зависимости для эмуляции и основные сценарии отказов.

Задание 4. Сети, Linux и Безопасность

- Работа с файловой системой (создание, копирование, права доступа chmod/chown), просмотр процессов (ps, top, kill).
- Найти причину недоступности учебного сервиса с помощью curl, dig, ss, lsof и анализа логов.
- Поиск информации в логах: даётся access.log — с помощью grep/awk/sed найти ошибки 5xx, топ IP-адресов, количество запросов по методам.
- Перехват и модификация трафика через снифферы: подмена ответа API, проверка поведения приложения.

Задание 5. Автоматизация тестирования API

- Реализовать API-автотесты на выбранном стеке:
  - Python: pytest + requests/httpx;
  - Java: JUnit/TestNG + REST Assured.
- Добавить проверки статуса, тела, заголовков, схемы ответа и очистку тестовых данных.
- Настроить/реализовать Mock для эмуляции успешного ответа, ошибки и задержки внешнего API.

Задание 6. Автоматизация тестирования UI (Playwright)

- Автоматизировать авторизацию и один основной пользовательский сценарий:
  - Python: Playwright + pytest;
  - Java: Selenide + JUnit/TestNG.
- Реализовать Page Object/Component Object, устойчивые локаторы и корректные ожидания.
- Добавить параметризацию и подготовку тестовых данных через API.
- Автоматизировать сбор диагностических материалов: trace, скриншоты и логи браузера.

Задание 7. Базы данных для тестировщиков

- Написание SELECT-запросов с JOIN, GROUP BY, подзапросами для проверки данных после выполнения тестового сценария.
- Вставка (INSERT), обновление (UPDATE) тестовых данных через SQL перед запуском автотестов, очистка после (teardown).
- Анализ плана запроса, предложения по тестированию.
- Проект тестирования миграции БД, с учетом роллбека и негативных сценариев.

Задание 8. Docker, CI/CD

- Создать отдельную Git-ветку, оформить коммиты и Merge Request с изменениями тестового проекта.
- Написание Dockerfile для тестового проекта, сборка образа, локальный запуск контейнера.
- Создание docker-compose.yml: поднять тестовую среду, дождаться готовности (healthcheck).
- Настройка CI/CD в GitLab CI: добавить статический анализа кода проекта с тестами (линтер).

Задание 9. AI-инструменты в работе тестировщика

- Установить OpenCode, подключить модель и открыть учебный проект.
- Написать собственный скилл для одной конкретной QA-задачи и продемонстрировать его работу.

Задание 10. Итоговая практика и защита проекта

- Изучить требования, архитектуру, UI, API и данные нового учебного приложения.
- Подготовить тест-план, анализ рисков, чек-листы и матрицу покрытия.
- Провести исследовательское тестирование и оформить найденные дефекты.

- Автоматизировать согласованный набор API- и UI-сценариев на выбранном языке.
- Добавить проверку данных, запуск тестов в Docker и CI.
- Провести регрессионный прогон и подготовить отчёт о качестве продукта с рекомендацией о релизе.

Список литературы

1. Бхаргава, А. Грокаем алгоритмы / А. Бхаргава ; пер. с англ. Е. Матвеева. — 2-е изд. — Санкт-Петербург : Питер, 2024.
2. Криспин, Л. Agile-тестирование : обучающий курс для всей команды / Л. Криспин, Д. Грегори ; [перевод с английского Е. Бугаевой, Е. Гандиной, О. Климовой, О. Колесникова]. — Москва : Манн, Иванов и Фербер, 2019.
3. Куликов, С. С. Тестирование программного обеспечения. Базовый курс / С. С. Куликов. — 3-е изд. — Минск: Четыре четверти, 2020.
4. Персиваль, Г. Python. Разработка на основе тестирования / Г. Персиваль ; пер. с англ. А. В. Логунова. — Москва : ДМК Пресс, 2018.
5. Фарли, Д. Современная программная инженерия : ПО в эпоху эджайла и непрерывного развертывания / Д. Фарли ; [пер. с англ. М. Трусковская]. — Санкт-Петербург [и др.] : Питер, 2023.
6. Уиттакер, Дж. Как тестируют в Google / Дж. Уиттакер, Дж. Арбон, Дж. Каролло ; [перевели с англ. А. Васюхина, Ю. Нечаева]. — Санкт-Петербург : Питер, 2014.

Интернет-источники

1. https://vladislaveremeev.gitbook.io/qa_bible
2. https://software-testing.ru/
3. https://learngitbranching.js.org/
4. https://sql-academy.org/ru/trainer