ГЛАВА 1

Введение в фулстек-тестирование

В современном мире цифровизация необходима для поддержания и развития любого бизнеса. Многие компании ушли далеко вперед в этом направлении, но есть и такие, которые только начинают строить свои цифровые платформы.

Цифровизация помогает бизнесу расширить охват потребителей от местного их круга до глобального масштаба, что способствует росту производства и увеличению доходов. Почти все малые и крупные предприятия в различных секторах экономики, таких как здравоохранение, розничная торговля, туризм, наука, социальные сети, банковское дело и развлечения, расценивают продвижение своих цифровых стратегий как важнейшую меру, предпринимаемую для охвата новых групп клиентов и получения более высокой прибыли.

На этом пути к цифровизации и модернизации решающим фактором становятся инновации. Предприятия, постоянно внедряющие инновации, остаются сильными и процветают на протяжении многих десятилетий. Классический пример — компания Netflix. В самом начале, в 1990-х годах, это был портал онлайн-проката DVD, а в 2007-м она сосредоточилась на онлайн-трансляции, поглотив собственный бизнес по прокату DVD. Позже она начала производить оригинальный контент под названием *Netflix Originals*. По состоянию на конец 2021 года Netflix была крупнейшим онлайн-сервисом потокового вещания (https://oreil.ly/AyHBL) с более чем 200 млн подписчиков по всему миру.

Параллельно с инновационным бизнесом развивается и технологическое пространство, стремясь удовлетворить растущие потребности. Прошли те времена, когда люди были готовы стоять в очереди, чтобы купить билеты в кино, ехать в отдаленный магазин, чтобы приобрести специфический продукт, или носить с собой написанный от руки список покупок, чтобы ничего не забыть. Технологии облегчают решение таких повседневных задач. Мы можем сидеть дома, смотреть любимые развлекательные программы и одним нажатием кнопки виртуально примерить новое платье, запланировать доставку продуктов, сварить кофе с помощью голосовой команды и сделать многое другое.

Учитывая быстрые темпы развития технологий, стратегии цифровизации должны быть универсальными, удовлетворять различные потребности клиентов и позволять оставаться конкурентоспособными в данном секторе. Уже недостаточно просто создать веб-сайт — нужно что-то более масштабное. Рассмотрим такие компании, предоставляющие услуги такси, как Uber и Lyft. Они позволяют получить доступ к своим услугам различными способами: через Интернет, мобильные платформы Android и iOS и даже чат-бот WhatsApp (https://oreil.ly/1ijA9). Универсальная стратегия цифровизации помогла этим компаниям распространиться по всему миру и перерасти своих конкурентов.

Инновации и гибкость помогают предприятиям добиться успеха в привлечении критической массы клиентов. Но после этого встает задача обеспечения дальнейшего процветания, получения бóльших доходов и привлечения еще большего числа клиентов. Мы видели, как отраслевые гиганты, такие как Amazon, используют свою клиентскую базу для организации перекрестных продаж товаров и услуг и дальнейшего расширения бизнеса. Компания Amazon, которая на первых порах существовала как книжный интернет-магазин, теперь занимается перекрестными продажами товаров от продуктов питания до электроники, одежды, ювелирных изделий и многого другого, удовлетворяя спрос на потребительские товары практически во всех сегментах рынка.

Почему я упоминаю эти детали в книге по тестированию программного обеспечения? Потому что современная индустрия программного обеспечения удовлетворяет все эти потребности бизнеса, предоставляя инструменты для выработки новых идей, воплощения их в жизнь и масштабирования с целью охвата новых групп клиентов по всему миру. Несомненно, команды разработчиков ПО находятся на переднем крае, особенно когда срочно требуется обеспечить *высокое качество*! Действительно, качество ПО стало важным критерием на современном конкурентном рынке. Компромиссы в отношении качества ПО эквивалентны признанию проигрыша в конкурентной гонке, что подтверждается многими реальными примерами. Например, в октябре 2014 года индийские гиганты электронной коммерции Snapdeal и Flipkart после многомесячной рекламной кампании провели сезонную распродажу. К сожалению, во время нее, в День большого миллиарда, веб-сайт Flipkart (https://oreil.ly/C20pD) несколько раз падал из-за огромного наплыва посетителей, в результате чего компания потеряла массу потенциальных клиентов и недосчиталась значительного объема доходов. Точно так же компания Yahoo! не смогла на равных состязаться со своими конкурентами, несмотря на то что была одной из первых в своем сегменте. Случилось это отчасти потому, что ее специалисты не уделили должного внимания качеству поисковой системы (https://oreil.ly/CiYDd), а отчасти из-за ущерба, нанесенного бренду недостаточными мерами безопасности. Это привело к крупнейшей утечке данных в истории (https://oreil.ly/CP5ma), в результате которой в 2013 году были раскрыты 3 млрд учетных записей пользователей. Такова цена качества ПО в наши дни!

Подобные примеры можно найти по всему миру. Они подтверждают наблюдение о том, что предприятия, несмотря на любые новые идеи, оказываются на крутом и скользком склоне, когда качество ставится под угрозу, потому что клиенты быстро переходят к более надежным конкурентам. Иногда компании вынуждены выходить на рынок как можно раньше, несмотря на недостаточно высокое качество их программного продукта, но они должны осознавать, что тем самым создают себе технический долг, который необходимо погасить раньше, чем конкуренты воспользуются им в своих интересах. Таким образом, можно утверждать, что качество — это важнейшее условие поддержания бизнеса в долгосрочной перспективе, а высокого качества можно достичь, только сочетая профессиональную разработку и тщательное тестирование и уделяя должное внимание каждому аспекту приложения. Чтобы помочь вам встать на этот путь, расскажу в этой главе, что входит в комплексное тестирование типичного мобильного или веб-приложения.

Фулстек-тестирование для достижения высокого качества

Для начала разберемся, что означают слова *«качество ПО»*. Когда-то под качеством ПО подразумевалось отсутствие ошибок, но сейчас любой причастный к созданию ПО согласится, что простого отсутствия ошибок уже недостаточно, чтобы назвать ПО качественным. Если попросить конечных пользователей дать определение понятия «качество», то вы услышите в ответ перечисление таких характеристик, как простота использования, привлекательный внешний вид, надежная защита персональных данных, высокая скорость предоставления информации и круглосуточная доступность услуг. Если попросить сделать то же самое компании, то вам расскажут о рентабельности инвестиций, аналитике в реальном времени, нулевом времени простоя, отсутствии привязки к поставщику, масштабируемой инфраструктуре, безопасности данных, соблюдении законодательства и о многом другом. Все эти аспекты определяют качество современного программного обеспечения. Недостатки в любой из этих областей так или иначе влияют на качество, поэтому их необходимо тщательно проверять!

Список требований к качеству выглядит длинным, но у нас есть инструменты и методологии, способные помочь удовлетворить большинство этих требований. Поэтому для достижения высокого качества необходимо знать эти инструменты и, что более важно, уметь применять их в контексте разработки и тестирования. Цель этой книги — помочь вам обрести навыки тестирования, необходимые для создания высококачественных мобильных и веб-приложений.

Если говорить кратко, то тестирование — это практика проверки того, соответствует ли поведение приложения нашим ожиданиям. Чтобы добиться успеха, тестирование необходимо практиковать на микро- и макроуровнях. Оно должно

быть неразрывно связано со всеми деталями приложения. Тестировать нужно каждый метод во всех классах, каждое значение во входных данных, каждое сообщение в логах, каждый код ошибки и т. д. Не менее важно сосредоточиться на макроаспектах, таких как тестирование возможностей и интеграция возможностей, а также сквозное тестирование рабочих сценариев. Но этим все не ограничивается! Необходимо дополнительно протестировать нефункциональные аспекты приложения — безопасность, производительность, доступность, удобство использования и т. д., чтобы достичь конечной цели — создать высококачественное ПО. Если кратко: мы должны провести *полноценное фулстек-тестирование*! Как показано на рис. 1.1, оно включает в себя тестирование не только различных аспектов приложения на каждом уровне (база данных, сервисы и пользовательский интерфейс), но и приложения в целом.

Рис. 1.1. Обзорная картина фулстек-тестирования

Фулстек-тестирование и разработка должны быть неотделимы друг от друга, как два рельса железнодорожного полотна. Чтобы обеспечить высокое качество продукта, мы должны двигаться в обоих направлениях сразу, иначе неизбежно сойдем с рельсов. Например, представьте, что вы пишете небольшой блок кода для приложения электронной коммерции, вычисляющий общую сумму заказа. Следует убедиться, что код правильно вычисляет сумму, выполняется безопасно и параллельно. Если этого не сделать, в железнодорожном полотне могут возникнуть разрывы, и, продолжив разработку, двигаясь по такой фрагментарной линии, вы получите плохую интеграцию и неоптимальную функциональность. Чтобы внедрить тестирование на таком элементарном уровне, командам необходимо перестать думать о нем как об особых действиях, выполняемых после разработки, как было принято раньше. Фулстек-тестирование должно проводиться параллельно с разработкой и практиковаться на протяжении всего цикла поставки, чтобы обеспечить быструю обратную связь. Практика, при которой тестирование начинается на ранних стадиях цикла поставки, называется *ранним тестированием*, или *сдвигом тестирования влево*, и это ключевой принцип, которому нужно следовать при выполнении фулстек-тестирования, чтобы получить хорошие результаты.

Тестирование на ранних этапах разработки ПО

Последовательность действий в традиционном жизненном цикле разработки ПО включает в себя анализ требований, проектирование, разработку и тестирование, причем последнее — в самом конце. Как видно на рис. 1.2, смещение тестирования на ранние этапы разработки ПО для получения высококачественных результатов предполагает перенос действий по тестированию в начало цикла.

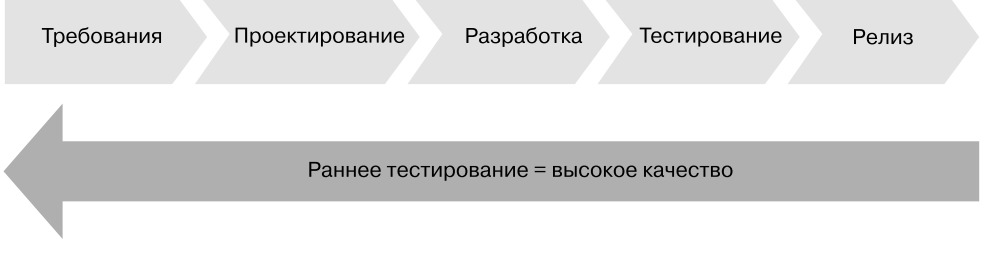

Рис. 1.2. Тестирование на ранних этапах разработки ПО

Рассмотрим аналогию, чтобы лучше понять этот принцип. Представьте, что ваша команда строит дом. Разумно ли сначала завершить строительство и только потом проверять качество? А что, если вы обнаружите, что размеры комнат не соответствуют проекту или внутренние стены недостаточно прочны, чтобы выдержать

нагрузку? Избежать именно таких проблем помогает смещение тестирования на ранние этапы разработки ПО, то есть проверка качества на этапе планирования и на протяжении всего строительства. Такой подход позволяет добиться максимально высокого качества конечного продукта.

Выполнение проверок качества на протяжении всего этапа разработки означает их итеративное повторение после завершения каждого небольшого фрагмента работы, чтобы вносимые изменения не ухудшали качество продукта. В аналогии со строительством дома это означает проверку при возведении каждой стены, чтобы можно было немедленно устранять любые огрехи и недоделки. Для выполнения таких обширных проверок раннее тестирование в значительной степени опирается на автоматизированное тестирование и методы непрерывной интеграции и непрерывного развертывания (CI/CD), когда проверки качества автоматизированы на микро- и макроуровнях и постоянно реализуются для каждого небольшого фрагмента работы на сервере непрерывной интеграции. Только автоматизация может гарантировать постоянное тестирование приложения с минимальными затратами и усилиями по сравнению с ручным тестированием различных аспектов качества.

Чтобы понять, что это означает в контексте ПО, разобьем раннее тестирование на повседневные действия. Рассмотрим команду разработчиков, которая придерживается итеративного цикла разработки, принятого, например, в Agile-разработке. Некоторые проверки качества, которые они могут выполнять на разных этапах, показаны на рис. 1.3.

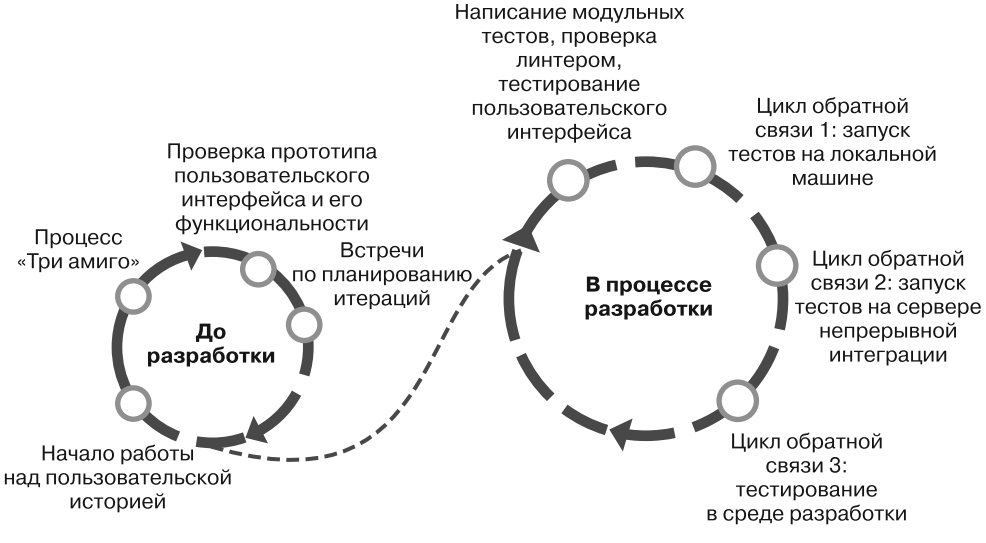

Рис. 1.3. Проверки качества, выполняемые при тестировании на ранних этапах разработки ПО

Диаграмма на рис. 1.3 начинается *(слева)* с определения набора проверок качества, которые команда выполняет до того, как пользовательскую историю можно будет считать готовой к разработке.

- На этапе анализа проходит процесс под названием «Три амиго» (https://oreil.ly/WFABh). В ходе него представители бизнеса, разработчики и тестировщики ненадолго собираются вместе и обсуждают функцию, предлагаемую к реализации. Этот процесс направлен на определение точек зрения всех трех ролей, чтобы не упустить из виду интеграцию, крайние случаи и другие бизнес-требования. Это первый шаг к раннему тестированию, когда с самого начала проверяются требования к функции.
- Одновременно представитель бизнеса и дизайнер пользовательского опыта (UX) совместно проектируют и улучшают дизайн приложения.
- По завершении этих двух шагов в начале итерации/спринта проводится *встреча по планированию итерации* (iteration planning meeting, IPM), на которой подробно обсуждаются пользовательские истории, которые предполагается реализовать на этой итерации. Это дает команде возможность еще раз коллективно проверить требования.
- Во время итерации непосредственно перед тем, как пользовательская история будет принята к разработке, начинается *работа над пользовательской историей* — уменьшенная версия процесса «Три амиго», в которой обсуждение фокусируется на требованиях и крайних случаях конкретной пользовательской истории. На этом этапе мы можем с уверенностью сказать, что команда тщательно протестировала/проверила требования.

Точно так же в ходе реализации пользовательской истории внедряются и задействуются следующие проверки качества, обеспечивающие быструю обратную связь.

- Разработчики пишут модульные тесты для каждой истории, добавляют линтеры и плагины для статического анализа кода и интегрируют их в конвейер непрерывной интеграции (CI) для получения постоянной обратной связи.
- В некоторых командах разработчики также пишут функциональные тесты для проверки пользовательского интерфейса и интегрируют их в конвейер CI. В других командах тестировщики пишут такие тесты уже после разработки. Оба подхода считаются обычной практикой.
- Прежде чем отправлять изменения в репозиторий, разработчики запускают набор автоматических тестов на своих локальных компьютерах, создавая первый уровень обратной связи.
- Второй уровень обратной связи подразумевает набор автоматизированных тестов (модульных, сервисов, пользовательского интерфейса и т. д.), которые запускаются на этапе непрерывной интеграции после каждой отправки кода в репозиторий.
- Третий уровень обратной связи образуется в результате процесса, называемого *тестированием в среде разработки*, когда тестировщики и представители

бизнеса проводят быстрый раунд ручного исследовательского тестирования на сервере разработки, чтобы быстро проверить недавно разработанную функциональность.

Благодаря такому пристальному вниманию к обратной связи команда выявит почти половину проблем, которые были бы обнаружены в ходе ручного тестировании после разработки, еще до того, как пользовательская история доберется до самой фазы тестирования. Другими словами, команда просто начинает тестирование на ранних этапах разработки ПО, давая тестировщикам полную свободу исследования пользовательской истории на предмет различных аспектов качества вместо проверки ожидаемого функционального поведения.

Таким образом, раннее тестирование позволяет не только предотвращать (за счет нескольких раундов проверки требований), но и обнаруживать любые дефекты, которые неизбежно появляются на ранней стадии, либо на локальной машине разработчика, либо на сервере CI. Кроме того, такой подход предоставляет тестировщикам возможность глубже изучить различные аспекты качества и тем самым гарантирует поставку высококачественного программного обеспечения.

Экстремальное программирование (extreme programming, XP) — это разновидность Agile-разработки программного обеспечения, которая предполагает раннее тестирование. Желающим глубже изучить методологии и практики XP я могу порекомендовать книгу Кента Бека Extreme Programming Explained[1] (Addison-Wesley Professional).

Идея включения тестирования в цикл поставки на ранних стадиях не ограничивается функциональным тестированием приложений. Она вполне применима к тестированию в целом, в том числе к тестированию безопасности, производительности и многим другим видам тестирования. Например, один из многих способов начать тестирование безопасности на ранних этапах — использовать инструмент сканирования перед отправкой кода в репозиторий, такой как Talisman, который проверяет код на наличие уязвимой информации и сообщает о найденных проблемах. В следующих главах вы увидите, как на практике происходит раннее тестирование.

В целом этот подход воплощает в себе девиз: «Качество — это ответственность команды», потому что проверки качества на каждом этапе жизненного цикла разработки программного обеспечения — прототипов приложений, требований и т. д., как говорилось ранее, — должны проводить разные члены команды. Таким образом, можно утверждать, что развитие навыков выполнения различных проверок качества у всех членов команды имеет решающее значение для создания высококачественного ПО!

---

[1] *Бек К.* Экстремальное программирование: разработка через тестирование. — СПб.: Питер, 2024.

Десять навыков фулстек-тестирования

Размышляя о навыках тестирования, мы склонны объединять их в две широкие категории — ручное и автоматизированное тестирование. Но за последние несколько десятилетий технологии развились, и в эти широкие категории были включены новые важные навыки, которым необходимо обучаться для выполнения различных проверок качества и создания высококачественных мобильных веб-приложений. На рис. 1.4 приведены десять навыков фулстек-тестирования, которые позволяют эффективно выполнять его.

Рис. 1.4. Десять навыков фулстек-тестирования, владеть которыми обязательно для создания высококачественных мобильных веб-приложений

Рассмотрим эти десять навыков и узнаем, почему их стоит освоить.

Ручное исследовательское тестирование

Для начала уточним: ручное исследовательское тестирование отличается от ручного тестирования. Последнее относится к проверке заданного списка требо-

ваний и не всегда требует аналитического мышления. Ручное исследовательское тестирование, напротив, — это умение вникать в детали приложения, придумывать различные сценарии применения, помимо тех, что задокументированы в пользовательских историях, моделировать их в тестовой среде и наблюдать за поведением приложения. Этот навык требует логического и аналитического мышления и является первым и главным навыком, необходимым для создания качественных приложений. Существуют разные методологии и подходы к структурированию исследовательских сессий. Мы обсудим их в главе 2.

Автоматизированное функциональное тестирование

Это один из основных навыков, необходимых для раннего тестирования. Автоматизация тестирования значительно сокращает ручной труд, что особенно важно, когда приложение расширяется и в него включается все больше функций. Проще говоря, этот навык заключается в написании кода для автоматизированного тестирования требований к функциям, протекающего без вмешательства человека. Для этого нужны инструменты, поэтому для приобретения данного навыка требуется знание различных инструментов, которые можно использовать при написании тестов для разных уровней приложения. Однако на этом дело не заканчивается — необходимо также знать, какие антипаттерны следует искать при автоматизированном тестировании и как их избегать. Различные аспекты этого навыка обсудим в главе 3.

Непрерывное тестирование

Непрерывная доставка — это практика постепенной доставки функций конечным пользователям в ходе коротких циклов, а не в рамках одного масштабного выпуска. Благодаря непрерывной доставке компания начинает получать прибыль уже на раннем этапе и может быстро оценить и пересмотреть свою стратегию на основе отзывов конечных пользователей. Для непрерывной доставки необходимо постоянно тестировать приложение, чтобы оно всегда было готово к выпуску. Самый простой способ добиться этого — автоматизировать и интегрировать проверки качества в конвейеры CI/CD и запускать их как можно чаще, чтобы упростить тестирование. Навыки непрерывного тестирования включают в себя определение типов автоматических тестов, которые следует запускать на каждом этапе цикла доставки, чтобы команда могла эффективно интегрировать их в конвейеры CI/CD и быстрее получать обратную связь. Об этом подробно рассказывается в главе 4.

Тестирование данных

Возможно, вы слышали такие выражения: «Данные — это деньги» и «Данные — это новая нефть». Они подчеркивают, насколько в настоящее время важно тестировать целостность данных. Когда приложение теряет или показывает неверные данные, пользователи утрачивают доверие к нему. Навыки тестирования данных требуют знания различных типов систем хранения и обработки данных, обычно применяемых в мобильных и веб-приложениях (базы данных, кэши, потоки событий и т. д.), а также умения создавать подходящие тестовые

сценарии. Подробнее об этих темах, а также о том, как на основе потоков данных между компонентами приложения создавать новые тестовые сценарии, мы поговорим в главе 5.

Визуальное тестирование

Внешний вид приложения — один из основных факторов, влияющих на ценность бренда, особенно когда речь идет о крупных продуктах, реализующих взаимодействие бизнеса с потребителями и применяемых миллионами. Непривлекательный внешний вид может отрицательно повлиять на ценность бренда. Поэтому важно убедиться, что конечные пользователи получают гармоничный и приятный визуальный опыт, и для этого проводить визуальное тестирование приложения. Для эффективного визуального тестирования важно понимать, как компоненты пользовательского интерфейса взаимодействуют друг с другом и с браузером (в случае веб-приложений). Тестирование этого вида тоже можно автоматизировать с помощью инструментов, отличных от тех, что применяются для автоматизации функционального тестирования. Об этом навыке и различиях между этими двумя типами автоматизации мы поговорим в главе 6.

Тестирование безопасности

Нарушения в системе защиты давно стали притчей в современном мире. И даже такие гиганты, как Facebook и Twitter, не являются исключениями. Проблемы защиты дорого обходятся и конечным пользователям, и бизнесу с точки зрения потери или раскрытия конфиденциальной информации, юридических санкций и репутации бренда. Прежде тестирование безопасности считалось узкоспециализированным навыком, и квалифицированных тестировщиков этого профиля обычно привлекали лишь к концу цикла разработки, чтобы с их помощью отыскать проблемы безопасности. Но из-за нехватки профессионалов в сфере тестирования безопасности и растущего числа случаев взлома защиты командам разработчиков программного обеспечения рекомендуется включать базовое тестирование безопасности в повседневную работу. В главе 7 мы поговорим о том, как научиться мыслить подобно хакеру и выявлять проблемы безопасности в приложениях, а также обсудим инструменты для автоматизации сканирования системы защиты.

Тестирование производительности

Даже незначительное снижение производительности может привести к огромным финансовым и репутационным потерям — вспомните пример Flipkart. В число навыков тестирования производительности входит умение измерять набор ключевых показателей производительности на разных уровнях приложения. Тесты производительности тоже можно автоматизировать и интегрировать в конвейеры CI для получения постоянной обратной связи. Стратегию раннего тестирования производительности и соответствующие инструменты мы обсудим в главе 8.