

ЗАХАРОВ ВИКТОР

ТЕСТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ



ЗАНИМАТЕЛЬНАЯ ПРАКТИКА

Виктор Захаров

**Тестирование
программного обеспечения.
Занимательная практика**

«Автор»

2025

Захаров В. В.

Тестирование программного обеспечения. Занимательная практика
/ В. В. Захаров — «Автор», 2025

Погрузитесь в увлекательный мир практики тестирования программного обеспечения с книгой, которая является логическим продолжением издания по теории тестирования. Автор приглашает вас в захватывающее путешествие, где вы освоите анализ требований, проектирование тестов, создание тестовых артефактов и многое другое. Вас ожидает более 100 практических заданий и примеров из реальной жизни, которые помогут закрепить материал и поднять уровень ваших компетенций. Эта книга — настоящий практикум, полный вызовов и готовых решений, который станет надёжным фундаментом для вашей успешной карьеры в области тестирования.

© Захаров В. В., 2025

© Автор, 2025

Содержание

Предисловие	5
Практика в теории	6
Тестирование надписей	7
Тестирование полей для ввода текста	9
Тестирование кнопок	12
Тестирование радиокнопок	14
Тестирование чекбоксов	17
Тестирование ссылок	19
Тестирование выпадающих списков	20
Тестирование форм отправки данных	23
Тестирование форм поиска и результатов поиска	25
Тестирование вкладок	29
Тестирование меню приложений	32
Тестирование меню навигации сайтов	34
Тестирование изображений	36
Тестирование слайдеров и каруселей	38
Тестирование прогресс-баров	41
Тестирование видео- и аудиоплееров	43
Тестирование модальных окон	45
Тестирование всплывающих подсказок	47
Тестирование календарей и элементов выбора даты	49
Тестирование пагинации	52
Тестирование загрузки файлов	54
Конец ознакомительного фрагмента.	56

Виктор Захаров

Тестирование программного обеспечения. Занимательная практика

Предисловие

Представьте, что теория – это детальная карта, а практика – это реальное путешествие по указанным на ней маршрутам. Только в пути вы приобретаете уникальный опыт, который невозможно получить, просто разглядывая карту в уюте дома. Из этого следует, что без применения теории на практике невозможно набраться необходимого опыта для повышения квалификации, развития профессиональных навыков и углубления знаний. Эта книга поможет вам приблизиться к глубокому пониманию практической стороны тестирования программного обеспечения.

Данная книга является логическим продолжением издания «Тестирование программного обеспечения. Основы»¹, в котором изложены фундаментальные теоретические аспекты тестирования. Здесь мы сосредоточимся на практическом применении ранее рассмотренной теории. Рекомендуется сначала ознакомиться с предыдущей книгой, поскольку она содержит базовые знания, необходимые для эффективного решения задач. Без этой подготовки выполнение практических заданий может вызвать значительные трудности.

¹ Информация о книге «Тестирование программного обеспечения. Основы» доступна на сайте автора.

Практика в теории

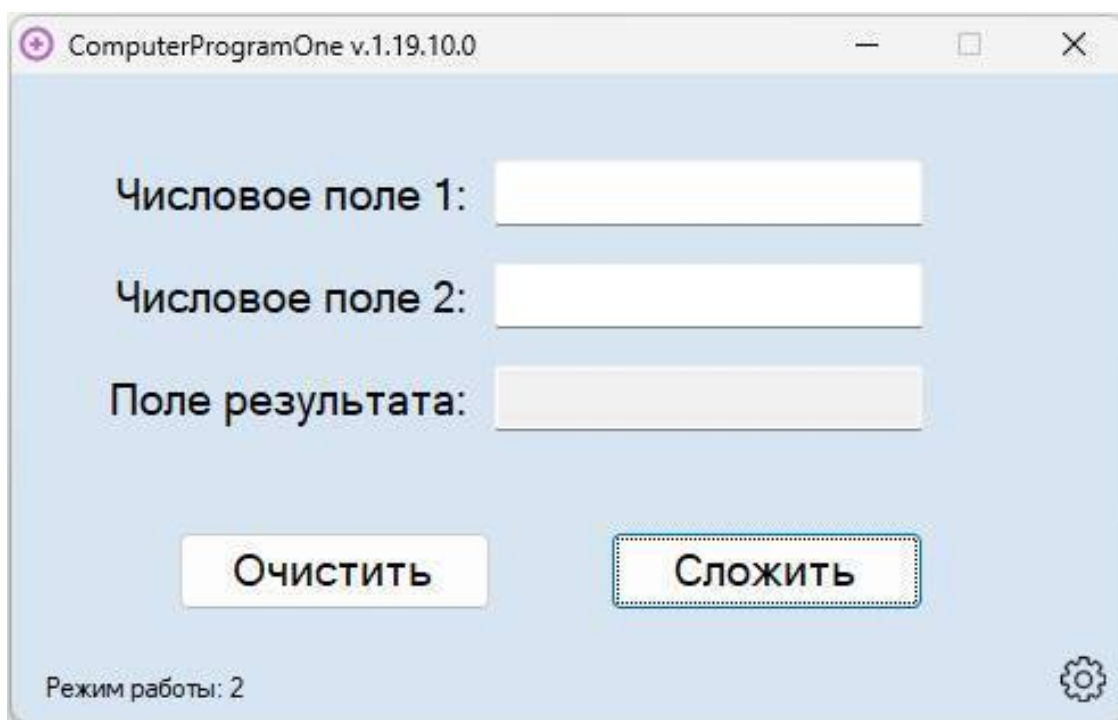
Прежде чем приступать к тестированию любого элемента, необходимо глубоко понимать сам процесс. Без достаточного опыта специалист рискует упустить ключевые аспекты, требующие особого внимания. Например, перед проверкой поля ввода текста в приложении тестировщику важно знать, как правильно подходить к этой задаче. Если у него нет практического опыта, он может не осознавать, на что именно следует обратить внимание. Эта глава посвящена тому, как применять теоретические знания и навыки на практике. Мы разберём основные элементы и базовые проверки, используемые при тестировании приложений².

В этой главе мы предлагаем практические рекомендации, которые помогут в будущем успешно решать реальные задачи. Наша цель – не охватить все возможные аспекты, а сосредоточиться на ключевых, фундаментальных элементах тестирования приложений. Поэтому мы не стремимся к исчерпывающему описанию, а акцентируем внимание на базовых компонентах и проверках.

² Под приложением подразумевается программа, сайт или мобильное приложение.

Тестирование надписей

Тестирование надписей в интерфейсе приложений является важной частью обеспечения качества приложений и удобства их использования. Надписи должны быть чёткими, понятными и корректными, чтобы пользователи могли легко ориентироваться в интерфейсе приложения. При тестировании необходимо обращать внимание на все имеющиеся в интерфейсе надписи. Сколько надписей в интерфейсе данного приложения?



В интерфейсе приложения семь надписей, включая заголовок приложения и подписи кнопок (текст кнопок).

Рассмотрим основные аспекты, на которые следует обратить внимание при тестировании надписей.

Типографика и читабельность. При тестировании типографики и читабельности внимательно оцениваем ключевые аспекты, чтобы убедиться, что информация легко воспринимается пользователями. Сюда входит проверка размера шрифта, чтобы текст был достаточно большим для чтения без напряжения глаз, а также оценка цвета и контраста текста по отношению к фону, чтобы обеспечить хорошую видимость при различном освещении и на разнообразных устройствах. Межстрочный интервал и ширина символов также должны способствовать удобочитаемости текста, предотвращая слияние строк и упрощая сканирование текста глазами.

Правильность и однозначность. Необходимо удостовериться, что все надписи написаны без ошибок: орфографических, пунктуационных и грамматических. Это обеспечивает профессиональный вид приложения и способствует лучшему пониманию содержимого пользователями. Также важно проверить, что тексты ясны и не содержат двусмысленностей, чтобы избежать возможных недопониманий, которые могут повлиять на работу с приложением. Тщательное тестирование этих аспектов поможет убедиться, что информация передаётся чётко и корректно, что особенно важно для функциональных элементов интерфейса, таких как кнопки действий, подсказки и инструкции.

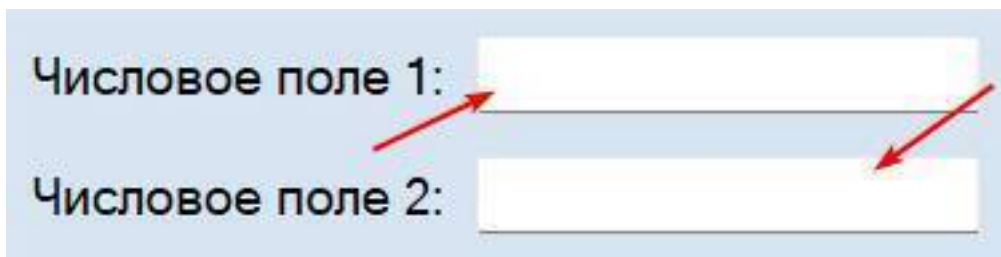
Соответствие контексту. При тестировании соответствия контексту необходимо убедиться, что тексты надписей адекватно отражают ситуацию их использования, обеспечивая чёткое и логичное взаимодействие пользователя с интерфейсом. Это особенно важно в диалоговых окнах, где тексты должны точно соответствовать доступным действиям пользователя, например, подтверждение удаления файла или сохранение изменений. Надписи должны быть сформулированы так, чтобы исключить любые возможности недопонимания и предоставить пользователю полную ясность в отношении того, какие действия будут выполнены при нажатии на кнопки.

Согласованность. Убеждаемся в единообразии надписей по всему интерфейсу, что важно для предотвращения путаницы. Необходимо проверить, чтобы одни и те же действия или объекты обозначались одинаковыми терминами в разных частях приложения. Например, если для отправки данных используется кнопка «Отправить» в одном разделе, то в других разделах не должно быть кнопок «Подтвердить» или «Запустить» для выполнения той же функции. Это помогает избежать ситуаций, когда пользователь затрудняется в понимании или даже делает ошибки из-за несогласованных обозначений.

Интерактивность. Необходимо удостовериться, что все интерактивные элементы, такие как ссылки или кнопки, чётко выделены и легко узнаваемы для пользователя. Ссылки должны отличаться от обычного текста по стилю – обычно они подчёркнуты или выделены цветом, чтобы пользователь мог сразу понять, что по ним можно кликнуть. Это обеспечивает интуитивное взаимодействие и помогает избежать путаницы, делая интерфейс дружелюбным и понятным для пользователей.

Тестирование полей для ввода текста

Тестирование полей для ввода текста – это проверка элементов интерфейса, в которые пользователи могут ввести информацию, такую как имя, пароль, комментарии, числовые данные и прочую информацию. Важно убедиться, что эти поля работают правильно и удобны для использования.



Обратим внимание на важные детали, касающиеся проверки полей для ввода текста.

Функциональность. При тестировании функциональности полей ввода основное внимание следует уделить проверке того, что поля корректно принимают и отображают введенные данные. Важно убедиться, что при клике на поле ввода появляется курсор, позволяя пользователю начать ввод текста. Также необходимо проверить, что текст вводится и отображается без ошибок или искажений, а также соответствует ожидаемому формату, например, для полей с ограничениями на количество символов или специфические форматы данных (как в случае с электронной почтой или телефонными номерами).

Ограничения на ввод. При анализе ограничений на ввод в полях формы важно проверить, что установленные ограничения точно соблюдаются. Это включает в себя проверку максимального и минимального количества символов, которые можно ввести в поле, а также ограничения на типы данных: например, проверьте, что поле, предназначенное только для ввода цифр, не принимает буквы или специальные символы. Если поле должно принимать комбинацию различных символов, убеждаемся, что оно корректно обрабатывает и отображает все разрешенные типы данных. Также важно проверить, что при попытке ввода недопустимых данных система корректно информирует пользователя о нарушении правил ввода, предоставляя ясные инструкции или сообщения об ошибке.

Проверка данных. Необходимо удостовериться, что поля корректно анализируют введенную информацию и реагируют на ошибки. Например, при вводе электронного адреса поле должно проверять, соответствует ли введенная строка формату адреса электронной почты и выдавать предупреждение, если формат некорректен. Сообщения об ошибках должны быть четкими и понятными, что поможет пользователю исправить ошибку без дополнительных затруднений. Эти тесты способствуют тому, что приложение не только предотвращает ввод неверных данных, но и помогает пользователям корректировать свои действия.

Копирование и вставка. При тестировании функциональности копирования и вставки необходимо убедиться, что поля ввода поддерживают эти операции. Необходимо проверить, можно ли копировать текст из полей и вставлять текст в них, а также удостовериться, что при вставке данные корректно отображаются и сохраняют свой первоначальный формат. Это особенно важно для полей, принимающих форматированный текст или специфические данные, такие как номера телефонов или адреса электронной почты. Также стоит проверить, что система правильно обрабатывает попытки вставить недопустимый или потенциально вредоносное содержимое, например, скрипты или слишком длинные строки, предотвращая возможные ошибки или угрозы безопасности.

Очистка полей ввода. При проверке возможности очистки полей ввода необходимо проверить, что пользователи могут удалять введённые данные с использованием клавиатурных клавиш, таких как «Backspace» и «Delete», а также через любые специальные кнопки очистки, предусмотренные в интерфейсе приложения. Эта проверка должна подтвердить, что после использования указанных методов поля действительно очищаются и готовы к новому вводу данных без оставшихся данных от предыдущего ввода.

Перенос строк. Нужно убедиться, что многострочные поля поддерживают перенос строк, что позволяет пользователям организовывать текст более чётко и понятно. Это особенно важно для текстовых полей, где требуется вводить длинные сообщения или комментарии, например, в формах обратной связи или текстовых редакторах. Необходимо удостовериться, что нажатие клавиши «Enter» приводит к созданию новой строки, а не к отправке данных формы, если иное не предусмотрено спецификой приложения. Также следует проверить, что переносы строк корректно сохраняются и отображаются при последующем просмотре или редактировании данных.

Ввод паролей. При тестировании полей для ввода паролей необходимо проверить, что введённые символы маскируются специальными символами (обычно это точки или звёздочки), чтобы обеспечить конфиденциальность данных пользователя. Это стандартная мера безопасности, предотвращающая возможность подглядывания пароля посторонними лицами во время ввода. Также следует проверить, что функция копирования и вставки отключена для полей пароля, чтобы минимизировать риск несанкционированного извлечения пароля. В дополнение к этому полезно удостовериться, что система предлагает надёжные меры по обработке и хранению паролей, такие как шифрование, чтобы усилить защиту пользовательских данных.

Десятичные числа. При проверке полей ввода, предназначенных для работы с десятичными числами, необходимо тщательно проверить их способность корректно обрабатывать и принимать числа с дробными. Важно удостовериться, что поле принимает как точку, так и запятую в качестве десятичного разделителя, если это поддерживается приложением, а также корректно обрабатывает введённые данные в обоих форматах. Особое внимание следует уделить проверке влияния системных настроек на ввод десятичных данных, так как в разных регионах могут использоваться различные стандарты для разделения целой и дробной части числа. Кроме того, необходимо проверить, как система обрабатывает сценарии, когда десятичная точка ставится в начале числа (например, «.5»), и убедиться, что такие числа корректно суммируются или обрабатываются в расчётах (например, «1.5 + .5» должно корректно рассчитываться как 2.0 или приложение должно сообщать об обязательном вводе ведущего нуля – «0.5»). Эти тесты помогут гарантировать, что поля ввода надёжно работают с числовыми данными в любых форматах и условиях, предоставляя точные и ожидаемые результаты.

Граничные значения. При тестировании граничных значений для полей ввода, где заданы ограничения на размер или определённые границы ввода данных, крайне важно провести тщательный анализ с использованием специализированных тестов для граничных значений. Проверяем минимально и максимально возможные вводимые значения, чтобы убедиться, что система корректно обрабатывает эти крайние случаи. Например, если поле ограничено 100 символами, следует проверить ввод 100 символов и убедиться, что они принимаются, а также попытаться ввести 101 символ, чтобы проверить, правильно ли система блокирует этот ввод. Аналогично следует проверить поведение поля при вводе минимально допустимого значения, например, одного символа или минимального числа в числовом поле. Такие тесты помогут обнаружить потенциальные ошибки в обработке данных на границах допустимых значений и гарантировать, что ввод пользователя будет обработан корректно.

Рассмотрим дополнительные аспекты для полей с ограниченным вводом.

Тип вводимых данных. При проверке полей с ограничениями на типы вводимых данных необходимо проверить, что механизмы приложения работают правильно. Это включает тести-

рование полей, предназначенных для ввода специфических данных, таких как даты, телефонные номера, номера кредитных карт или числа, чтобы убедиться, что они принимают только допустимые форматы и корректно обрабатывают ввод. Например, если поле предназначено только для ввода чисел, необходимо проверить, что при попытке ввести буквы или специальные символы данные не принимаются. Такие тесты помогают гарантировать, что ввод пользователя соответствует ожидаемым требованиям и что приложение исключает некорректный ввод, предотвращая возможные ошибки в обработке данных.

Формат ввода данных. Необходимо убедиться, что поля ввода строго соответствуют заданным форматам для определённых типов информации, таких как даты или телефонные номера. Это требует тщательного тестирования механизмов валидации, чтобы убедиться, что поля не принимают данные, которые не соответствуют ожидаемому формату. Например, если поле предназначено для ввода даты в формате «ДД.ММ.ГГГГ», следует проверить, что ввод любого отличающегося формата (например, «ММ/ДД/ГГГГ») вызывает ошибку валидации и предупреждает пользователя о некорректном вводе. Также стоит проверить, как приложение реагирует на ввод символов или неполных данных, чтобы убедиться в надёжности и точности проверок.

Автоматическое дополнение. В процессе тестирования функции автоматического дополнения в полях ввода важно оценить, насколько эффективно и точно она работает. Необходимо проверить, что при начале ввода система предлагает релевантные варианты для завершения введённых данных, например, предлагая закончить ввод адреса электронной почты после введения первых нескольких символов. Тестирование должно включать разные сценарии использования, чтобы убедиться, что предложения актуальны и соответствуют вводимым данным. Кроме того, важно проверить скорость реакции приложения на ввод пользователя, а также то, как оно обрабатывает неверный ввод или необычные варианты данных. Также следует удостовериться, что функция автоматического дополнения не нарушает приватность и безопасность данных пользователя, предлагая только те варианты, которые соответствуют ранее введённой информации.

Тестирование кнопок

Тестирование кнопок – это процесс, направленный на проверку их функциональности, визуального отображения, доступности и взаимодействия с пользователем.



Разберём основные элементы, имеющие значение при тестировании кнопок.

Возможность нажатия, активность. Необходимо убедиться, что каждая кнопка доступна для взаимодействия с ней. Проверяем, что кнопка реагирует на нажатия, проведённые различными методами ввода, такими как мышь, сенсорный экран и клавиатура, и что реакция на взаимодействие соответствует ожидаемому результату без задержек или ошибок. Проверки включают проверку визуальной обратной связи при нажатии, например, изменения цвета или анимации, подтверждающих активацию кнопки. Такие тесты помогут обеспечить надёжное и интуитивно понятное взаимодействие пользователей с приложением, повышая удобство и эффективность его использования.

Выполнение заданного действия. При проверке выполнения заданного действия каждой кнопкой важно убедиться, что она точно соответствует своему предназначению, как описано в требованиях к приложению. Необходимо проверить, что при нажатии кнопка выполняет именно то действие, которое от неё ожидается, будь то отправка формы, переход на другую страницу, обновление информации или любое другое действие. Кроме того, следует проверить, что действие выполняется без задержек и сопровождается соответствующими уведомлениями или изменениями интерфейса, подтверждающими успешное выполнение операции.

Состояния кнопки. При тестировании визуальных состояний кнопки важно убедиться, что каждое состояние – обычное, при наведении курсора, при нажатии, и неактивное, когда кнопка заблокирована – чётко различимо и корректно функционирует. Проверяем изменения цвета, подсветки или текстуры, которые должны наглядно информировать пользователя о возможности взаимодействия или его ограничениях. Также важно, чтобы после активации кнопки она возвращалась в исходное визуальное состояние, подтверждая завершение действия.

Подписи. Для обеспечения лучшей доступности и удобства использования каждая кнопка в интерфейсе должна быть снабжена чёткой подписью, которая точно описывает её функцию. Это важно, чтобы пользователи могли легко понять, какое действие будет выполнено при нажатии на кнопку, без необходимости догадываться или искать дополнительную информацию. Подписи должны быть краткими, но информативными, и использовать язык, понятный целевой аудитории.

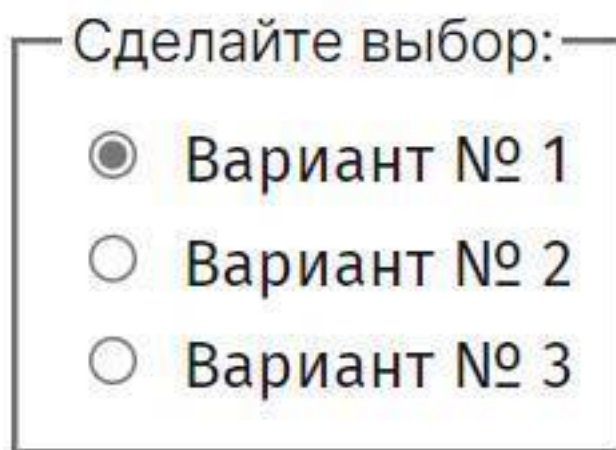
Защита от многократного нажатия. При проверке кнопок, которые инициируют действия, связанные с обработкой данных, такие как отправка формы, особое внимание необходимо уделить проверке защиты от многократного нажатия. Необходимо убедиться, что механизм блокировки кнопки или другие методы предотвращения повторной отправки данных работают корректно. Тестирование должно включать сценарии, когда пользователь пытается нажать кнопку несколько раз подряд, чтобы проверить, что данные не отправляются повторно и приложение обрабатывает только одно нажатие. Также стоит проверить, что кнопка корректно разблокируется или восстанавливает своё состояние после успешного завершения операции и

что пользователю предоставляется соответствующая обратная связь о статусе выполненного действия.

Подсказки на критические действия. Если кнопки выполняют важные или необратимые действия, такие как удаление данных, необходимо убедиться в наличии визуальных подсказок или запросов на подтверждение действия. Проверяем, что перед выполнением таких операций пользователю отображаются предупреждения или запросы на подтверждение, например, в виде всплывающих окон с вопросом «Вы уверены, что хотите удалить этот элемент?». Эти подсказки должны быть хорошо заметны и понятны, чтобы пользователь мог осознанно подтвердить или отменить действие. Тестирование должно охватывать различные сценарии, включая случайное нажатие кнопки, чтобы убедиться, что приложение надёжно предотвращает нежелательные или ошибочные операции.

Тестирование радиокнопок

Радиокнопка – это специальные кнопки, которые позволяют выбирать один из нескольких вариантов ответа. Когда нажимаем на радиокнопку, она светится или отмечается специальным образом, чтобы показать, что выбран именно этот вариант. Из нескольких радиокнопок можно выбрать только один вариант.



Тестирование радиокнопок в интерфейсе приложений включает проверку их функциональности, взаимодействия с пользователем, визуального представления и доступности. Проанализируем основные моменты, которые следует принимать во внимание при проверке радиокнопок.

Выбор по умолчанию. При тестировании радиокнопок проверяем, правильно ли установлен первоначальный выбор, если он предусмотрен требованиями. Это означает, что при открытии формы или страницы с набором радиокнопок должна быть выбрана именно та кнопка, которая указана в требованиях, или ни одна кнопка не должна быть выбрана, если такова задумка. Например, если по умолчанию должен быть выбран определённый вариант, убеждаемся, что он автоматически активен при загрузке страницы.

Сохранение выбора. Проверяем, что выбор пользователя сохраняется корректно, особенно при таких действиях, как перезагрузка страницы или отправка формы. Это означает, что после выполнения этих действий выбранная пользователем радиокнопка должна оставаться активной, сохраняя ранее сделанный выбор. Проверка этого аспекта гарантирует, что приложение сохраняет выбор пользователя без необходимости повторного выбора.

Отклик на действия пользователя. Необходимо удостовериться, что радиокнопки корректно реагируют на все способы взаимодействия, включая клики мышью, касания на сенсорных экранах и навигацию с клавиатуры. Радиокнопки должны незамедлительно активироваться при любом из этих действий, чётко отражая выбранный пользователем вариант.

Отличие состояний. При тестировании радиокнопок важно убедиться, что пользователь легко может отличить выбранное состояние от невыбранного. Это означает, что радиокнопка в активном состоянии должна визуально выделяться, например, заполненным кружком, цветом или другой заметной меткой, в то время как невыбранные кнопки остаются пустыми или менее выраженными. Такое различие помогает пользователю сразу понять, какой вариант выбран.

Группировка радиокнопок. Проверяем, что кнопки, относящиеся к одной группе, визуально сгруппированы вместе, чтобы пользователи могли легко понять, что из предложенных

вариантов можно выбрать только один. Эта группировка может быть достигнута с помощью расположения кнопок близко друг к другу, применения общего заголовка или рамки вокруг группы. Такое визуальное оформление помогает пользователю интуитивно понять, что все радиокнопки внутри этой группы связаны между собой, и выбор одного варианта автоматически исключает возможность выбора другого.

Группа № 1:

- ☒ Вариант № 1
- ☐ Вариант № 2
- ☐ Вариант № 3

Группа № 2:

- ☐ Вариант № 1
- ☒ Вариант № 2
- ☐ Вариант № 3

Единичный выбор. При проверке радиокнопок в группе необходимо убедиться, что приложение позволяет выбрать только один вариант. Это означает, что при выборе одной радиокнопки все другие кнопки в той же группе автоматически снимают свой выбор. Такой механизм гарантирует правильное функционирование радиокнопок, позволяя пользователю сделать только один выбор из предложенных вариантов, что является основным принципом их работы.

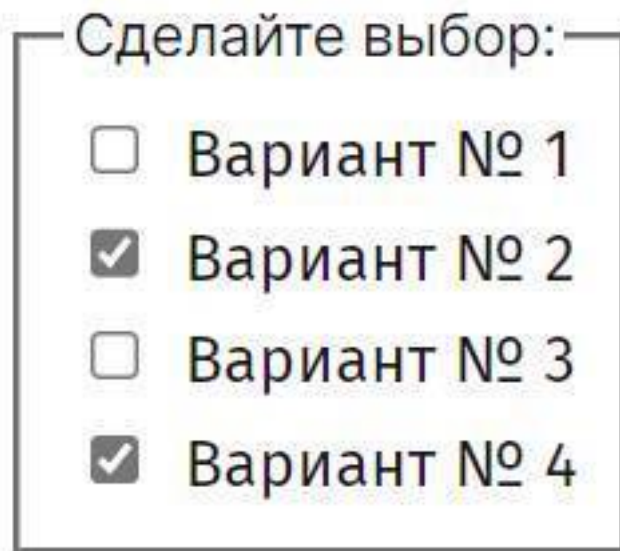
Взаимодействие с другими элементами. Проверяем, как выбор радиокнопок влияет на другие элементы интерфейса. Например, необходимо убедиться, что выбор определённой радиокнопки может активировать или деактивировать другие элементы, такие как кнопки, или привести к появлению дополнительных полей ввода, которые пользователю нужно заполнить. Это взаимодействие должно обеспечивать логичное и предсказуемое поведение интерфейса.

Проверка данных. При тестировании логики проверки данных в интерфейсе с радиокнопками необходимо убедиться, что эта проверка корректно работает, особенно в случаях, когда выбор одного из вариантов является обязательным. Это означает, что, если поль-

зователь не сделал выбор, система должна выдавать соответствующее предупреждение или не позволять продолжить выполнение действия, например, отправку формы, до тех пор, пока не будет выбрана одна из радиокнопок. Тестирование должно включать сценарии с отсутствием выбора, попыткой обойти обязательный выбор, а также с правильным выбором, чтобы подтвердить, что логика проверки данных надёжна и выполняется в соответствии с требованиями.

Тестирование чекбоксов

Чекбокс – это элемент интерфейса приложения в виде квадрата, где можно поставить или снять отметку (галочку). Когда чекбокс отмечен галочкой, это означает, что функция или опция активирована или выбрана. Например, когда оформляется заказ в интернет-магазине и видим чекбокс с надписью «Я соглашаюсь с условиями пользовательского соглашения», можем поставить галочку в этом чекбоксе, чтобы подтвердить своё согласие с условиями. Чекбоксы позволяют пользователям выбирать один или несколько вариантов из предложенного набора.



Рассмотрим, на что стоит обратить внимание при тестировании чекбоксов.

Изменение состояния. При проверке изменения состояния чекбоксов необходимо убедиться, что они корректно переключаются между состояниями «отмечено» и «не отмечено». Проверяем работу чекбоксов при клике мышью, касании на сенсорном экране и при навигации с помощью клавиатуры. Важно, чтобы при каждом из этих способов взаимодействия состояние чекбокса изменялось предсказуемо, без задержек и с правильным визуальным отображением выбранного состояния.

Множественный выбор. При тестировании чекбоксов необходимо убедиться, что они корректно поддерживают возможность множественного выбора, если это предусмотрено логикой работы приложения. Пользователь должен иметь возможность выбирать несколько опций одновременно, и все выбранные элементы должны сохраняться и правильно обрабатываться приложением. Если в интерфейсе представлено большое количество чекбоксов, необходимо проверить наличие и корректную работу механизма, позволяющего включить или отключить все чекбоксы одновременно, что значительно упрощает управление множественными опциями.

Чекбокс как радиокнопка. Если в интерфейсе чекбокс работает как радиокнопка, важно провести соответствующие тесты, чтобы убедиться, что его поведение соответствует ожиданиям для радиокнопок. В этом случае необходимо проверить, что при выборе одного чекбокса другие чекбоксы в той же группе автоматически снимают свой выбор, обеспечивая возможность выбора только одного варианта. Также следует проверить, что выбор чекбокса корректно сохраняется, состояние чётко отображается визуально и взаимодействие с другими элементами интерфейса происходит правильно.

По аналогии с радиокнопками проводим следующие проверки:

- Сохранение выбора.
- Отличие состояний.
- Взаимодействие с другими элементами.
- Проверка данных.
- Группировка чекбоксов.

Тестирование ссылок

Тестирование ссылок помогает обеспечить правильную работу приложения, повысить качество и удобство использования для конечных пользователей. Выделим главные аспекты, на которые стоит опираться в процессе тестирования ссылок.

Корректность URL³. При проверке работы ссылок важно удостовериться, что каждая ссылка ведёт на правильный URL-адрес, соответствующий её назначению. Проверяем, что URL-адреса, указанные в ссылках, корректны, не содержат опечаток и действительно приводят пользователя на ожидаемую страницу или ресурс. Ошибки в URL могут привести к появлению страниц с ошибками или отправке пользователя на неверный ресурс.

Атрибуты ссылок. Проверяем наличие и корректность атрибутов ссылок, таких как «title», «target», «rel» и других, в соответствии с требованиями, прописанными в требованиях. Атрибут «title» должен предоставлять дополнительную информацию, когда пользователь наводит курсор на ссылку, атрибут «target» определяет, будет ли ссылка открываться в новом окне или вкладке, а атрибут «rel» помогает определить отношения между текущей страницей и страницей, на которую ведёт ссылка. Важно убедиться, что все необходимые атрибуты присутствуют и настроены правильно. Рекомендую также ознакомиться с дополнительными материалами по атрибутам ссылок, чтобы глубже понять их значение и правильное использование.

Открытие ссылок. При тестировании открытия ссылок необходимо убедиться, что они корректно открываются при клике в соответствии с заданными настройками. Проверяем, что ссылки, настроенные с атрибутом «target», действительно открываются в соответствии с указанным параметром. Также важно проверить, что ссылки без этого атрибута открываются в том же окне, обеспечивая предсказуемое поведение для пользователя.

```
▼ <a target="_blank" href="https://victorz.ru/about" itemprop="url">  
  <span itemprop="name">Автор сайта</span>  
</a>
```

Визуальное отличие. Необходимо убедиться, что ссылки визуально отличаются от обычного текста, чтобы пользователи могли легко распознавать их как интерактивные элементы. Обычно это достигается с помощью цвета, подчёркивания или других стилевых решений, таких как изменение шрифта.

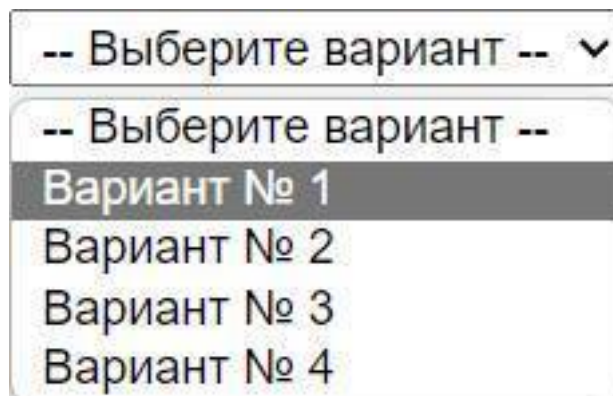
Изменение внешнего вида при наведении. Проверяем, что внешний вид ссылок изменяется при наведении курсора мыши, чётко указывая на интерактивность элемента. Это может включать изменение цвета, добавление или удаление подчёркивания, а также изменение вида курсора на указатель руки. Эти визуальные изменения помогают пользователю понять, что на элемент можно нажать, и подтверждают его интерактивность.

Понятные описания. Анализируя текст ссылок, обращаем внимание на то, чтобы они чётко и точно описывали, куда ведёт ссылка, избегаем неопределённых и общих фраз, таких как «нажмите здесь». Текст ссылки должен информировать пользователя о том, что он увидит или куда попадёт после клика, например, «Подробнее о наших услугах» или «Скачать руководство».

³ URL (Uniform Resource Locator, единый указатель ресурсов) – это адрес в интернете, который указывает на конкретное местоположение ресурса, такого как веб-страница, изображение, видеофайл и прочее. URL включает в себя имя веб-сайта (доменное имя) и путь к определённому ресурсу. Пример: <https://victorz.ru/about>.

Тестирование выпадающих списков

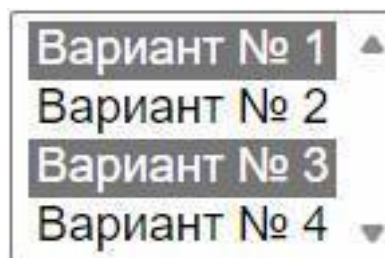
Выпадающий список – это элемент приложения, показывающий список опций, из которых пользователь может выбрать одну из опций. Когда кликают по данному элементу, он раскрывается, показывая все доступные для выбора варианты. Они полезны, когда нужно выбрать из нескольких предопределённых вариантов, например, страну проживания или цвет товара.



Рассмотрим аспекты, заслуживающие внимания при проверке выпадающих списков.

Открытие и закрытие списка. При тестировании выпадающих списков важно убедиться, что они корректно реагируют на различные виды взаимодействия, такие как клик мышью, сенсорный ввод или нажатие клавишей на клавиатуре. Список должен открываться без задержек и отображать все доступные опции. После того как пользователь выбирает нужный вариант или кликает вне списка, он должен автоматически закрываться, чтобы не мешать работе с другими элементами интерфейса.

Выбор вариантов. Проверяем, что пользователь может выбрать нужный ему вариант и что выбранный вариант корректно отображается в интерфейсе как выбранный. Для списков с одним вариантом нужно проверить, что после выбора этот вариант правильно отображается, заменяя текст по умолчанию. В случае выпадающих списков с поддержкой множественного выбора необходимо удостовериться, что пользователь может выбрать несколько опций, используя удерживание клавиши «Ctrl» (или аналогичное действие на других устройствах), и что все выбранные опции корректно отображаются. Также следует проверить, что система правильно обрабатывает и сохраняет этот множественный выбор, отображая его предсказуемым образом в интерфейсе:



Поиск и фильтрация. В выпадающих списках, поддерживающих поиск или фильтрацию вариантов, необходимо убедиться, что эти функции работают корректно и эффективно. Проверяем, что при вводе текста в поле поиска или фильтрации список автоматически обнов-

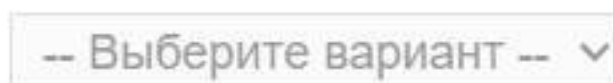
ляется, показывая только те варианты, которые соответствуют введенным данным. Необходимо убедиться, что поиск чувствителен к вводу, быстро реагирует и правильно отображает результаты, исключая неподходящие опции. Также следует протестировать различные сценарии, такие как ввод частичных совпадений или некорректных данных, чтобы убедиться, что система корректно обрабатывает все возможные ситуации.

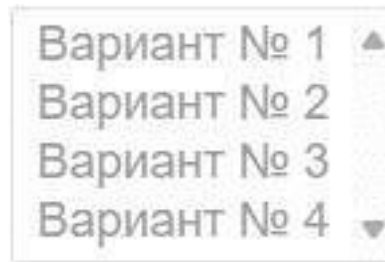
Группировка опций. При проверке выпадающих списков с группировкой опций важно убедиться, что группировка корректно отображается и помогает легко находить и выбирать нужные варианты. Проверяем, что все опции внутри группы логично организованы и визуально отделены от других групп, например, с помощью отступов, заголовков или разделителей. Группы должны быть ясно обозначены, чтобы пользователь мог сразу понять, к какой категории относятся предлагаемые варианты.



Визуальные подсказки. Необходимо убедиться в наличии визуальных подсказок, которые помогают пользователю понять, что список можно раскрыть. Такие подсказки могут включать стрелку вниз, расположенную рядом с полем, надпись «Выберите вариант из списка» или другой визуальный элемент, который указывает на возможность взаимодействия. Эти элементы должны быть хорошо заметны и интуитивно понятны, чтобы пользователи могли легко распознать, что перед ними выпадающий список, и знать, что им нужно сделать для его раскрытия.

Визуальное различие активного и неактивного состояний. Проверяем, что можно легко различить активные и неактивные состояния списка. Активный список должен быть визуально выделен, позволяя пользователю взаимодействовать с ним, выбирать опции и видеть доступные варианты. Неактивные списки, наоборот, должны быть четко обозначены как заблокированные для выбора – это может быть достигнуто с помощью изменения цвета, уменьшения яркости, добавления полупрозрачности или другого визуального эффекта, который ясно показывает, что этот список недоступен для взаимодействия:





Взаимодействие с другими элементами. Проверяем, как выбор вариантов в списке влияет на другие элементы интерфейса. Проверяем, что выбор определённого варианта может, например, активировать или деактивировать кнопки, показать или скрыть дополнительные поля ввода, или вызвать другие изменения в интерфейсе. Такие взаимодействия должны происходить плавно и предсказуемо, без задержек и с корректной реакцией приложения на действия пользователя.

Тестирование форм отправки данных

Форма отправки данных – это элемент приложения, в котором пользователь может ввести и отправить информацию. Это может быть что угодно: от простой формы обратной связи, где указываются имя и электронный адрес почты, до более сложных форм, таких как регистрация на сайте или заполнение онлайн-заявки. Формы могут включать в себя различные элементы, такие как текстовые поля, чекбоксы, радиокнопки, выпадающие списки и кнопки отправки.

Имя:

Фамилия:

Дата рождения:

Отправить

Рассмотрим тестирование элементов форм отправки данных.

Подсказки, подписи и инструкции. Необходимо проверить, что для каждого поля присутствуют чёткие подсказки, подписи и инструкции, которые помогают пользователю понять, какие данные нужно вводить и в каком формате. Это может быть текст рядом с полем, пример ввода или всплывающая подсказка. Такие элементы должны быть легко заметны и понятны, чтобы пользователь мог без затруднений заполнить форму корректными данными.

Проверка данных на стороне клиента⁴. Необходимо убедиться, что введённая информация правильно проверяется перед отправкой на сервер. Приложение должно проверять, что текстовые поля содержат допустимые данные, выбранные радиокнопки, чекбоксы, выпадающие списки сохраняют и передают корректные значения. Всё должно работать без ошибок, а при неправильном вводе пользователь должен получать понятные подсказки.

Сохранение данных. При тестировании формы проверяем, что введённые данные сохраняются в полях даже после того, как была обнаружена ошибка при проверке. Это означает, что, если пользователь допустил ошибку, ему не придётся вводить все данные заново – корректно заполненные поля должны сохранить своё содержимое. Такая функция значительно упрощает процесс исправления ошибок и повышает удобство работы с формой.

⁴ Клиент-серверная архитектура и многие другие моменты рассматриваются в книге «Тестирование программного обеспечения. Основы».

Отправка данных. При проверке отправки данных из формы важно убедиться, что все введённые и выбранные данные корректно отправляются на сервер и правильно обрабатываются. Проверяем, что сервер получает данные в ожидаемом формате и что они корректно записываются в базу данных⁵ или обрабатываются приложением. Особое внимание следует уделить формам, которые отправляют данные без перезагрузки страницы, используя асинхронную⁶ отправку. В этих случаях необходимо проверить, что процесс отправки происходит без ошибок, данные передаются полностью и корректно и пользователь получает соответствующую обратную связь, например, уведомление об успешной отправке или сообщение об ошибке, если что-то пошло не так.

Множественная отправка формы. Необходимо удостовериться, что приложение корректно обрабатывает ситуации с множественной отправкой данных. После успешной отправки формы приложение должно предотвращать повторную отправку тех же данных, например, путём очистки полей формы или блокировки кнопки отправки. Это предотвращает дублирование информации и возможные ошибки, связанные с множественной обработкой одного и того же запроса.

Быстродействие. При проверке формы необходимо оценить её быстродействие, чтобы убедиться, что отправка данных и проверка введённых данных на стороне клиента не вызывают заметных задержек в работе приложения. Пользователь не должен ощущать задержек или зависаний при взаимодействии с формой.

Проверка данных на стороне сервера. При тестировании проверки данных на стороне сервера важно убедиться, что все данные, отправленные через форму, проходят на сервере проверку на корректность. Также обращаем внимание на многострочные поля, чтобы убедиться, что они правильно обрабатывают переносы строк и сохраняют форматирование после отправки данных на сервер. Если сервер обнаруживает ошибки в данных, пользователю должно выводиться понятное и информативное сообщение, объясняющее, что именно пошло не так и как исправить проблему.

⁵ База данных (БД) – это организованная коллекция информации или данных, которая хранится в электронном виде и систематизирована таким образом, чтобы её можно было легко находить, управлять и обновлять. Обычно базы данных используются для хранения больших объёмов информации, которую компьютерные программы могут быстро обрабатывать и извлекать по запросу.

⁶ Асинхронная отправка данных формы – это процесс, при котором информация, введённая в форму в приложении, отправляется на сервер без необходимости перезагружать (обновлять) страницу приложения. Это значит, что после того, как пользователь нажимает кнопку «Отправить», данные из формы обрабатываются в фоновом режиме и пользователь может продолжать работу с приложением без прерывания.

Тестирование форм поиска и результатов поиска

Форма поиска – это элемент приложения, состоящий из поля для ввода текста, кнопки и, при необходимости, дополнительных элементов фильтрации, сужающих поиск. В форму поиска вводятся слова или фразы – запрос, чтобы найти нужную информацию. После ввода запроса и нажатия кнопки «Найти» или нажатия клавиши **Enter** на клавиатуре приложение просматривает доступные данные и отображает результаты, которые соответствуют введённому запросу.

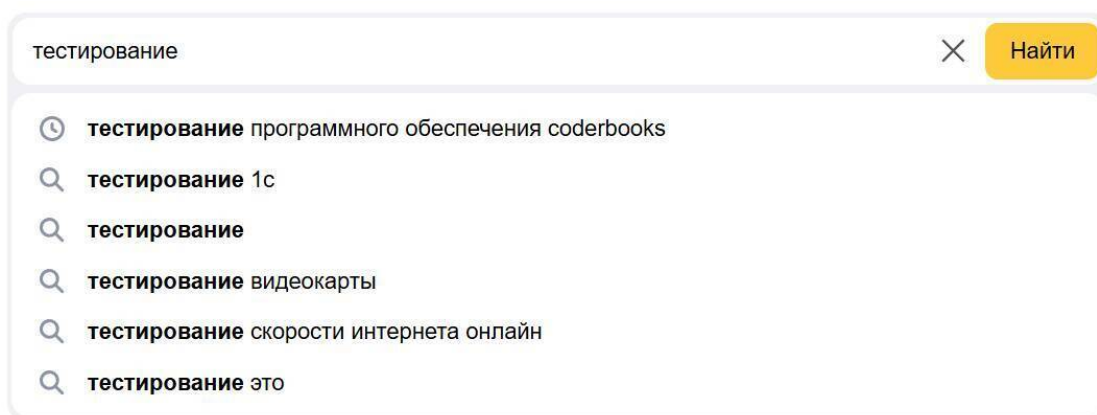


Форма поиска с дополнительными элементами фильтрации:



Подробно рассмотрим детали, на которые требуется обратить внимание при тестировании форм поиска.

Подсказки и автоматическое дополнение вводимых запросов. При тестировании форм поиска с функцией подсказок и автоматического дополнения вводимых запросов важно убедиться, что эти функции работают корректно и помогают пользователю в поиске. Подсказки и варианты автоматического дополнения должны точно соответствовать введённым данным, предлагая релевантные варианты, которые упрощают и ускоряют формирование запроса. Например, если пользователь начинает вводить название товара или ключевое слово, система должна предлагать популярные или подходящие варианты, позволяя пользователю выбрать из предложенного списка и тем самым быстрее найти нужную информацию. Важно также проверить, что предложенные подсказки и дополнения не вводят пользователя в заблуждение и не предлагают нерелевантные результаты:



Обработка пустых запросов. При анализе работы форм поиска важно уделить внимание обработке пустых запросов, когда пользователь пытается выполнить поиск, не введя никаких данных. Приложение должно корректно реагировать на такие ситуации, предотвращая выполнение бессмысленного поиска или предлагая полезные инструкции или рекомендации. Например, система может показать сообщение «Введите ключевое слово для поиска» или предложить популярные запросы, которые могут быть интересны пользователю. Такая обработка не только предотвращает ошибки, но и направляет пользователя.

Отображение статуса в процессе поиска. Необходимо убедиться, что пользователь получает своевременную и понятную обратную связь во время обработки запроса пользователя. Когда приложение выполняет поиск, должно быть ясно, что процесс продолжается – для этого часто используется индикатор загрузки⁷ или анимация, показывающая, что запрос обрабатывается. Такой визуальный сигнал информирует пользователя, что система работает над его запросом и помогает избежать неопределённости или повторных попыток запуска поиска. Это повышает уверенность пользователя в том, что его действие принято и скоро будет предоставлен результат.

Скорость поиска. Проверяем, что время, необходимое для обработки запроса и отображения результатов, должно быть минимальным и обеспечивать комфортное использование приложения. Пользователь не должен ощущать задержек, и результаты должны появляться практически мгновенно после ввода запроса. Помимо этого, необходимо протестировать производительность поисковой системы при одновременной обработке множества запросов от разных пользователей. Это позволяет убедиться, что приложение способно справляться с нагрузкой без снижения скорости и что даже при высоком трафике пользователи получают быстрые и точные результаты поиска.

Обработка специальных символов. При тестировании необходимо убедиться, что приложение корректно обрабатывает запросы, содержащие специальные символы или коды, такие как *, @, #, &, %, и другие. Эти символы не должны вызывать ошибок или ошибок в работе приложения. Вместо этого приложение должно либо корректно интерпретировать их в рамках поиска, либо игнорировать, если они не влияют на результаты. Например, если пользователь вводит запрос с символами «@фраза», система должна правильно обработать запрос и предоставить релевантные результаты без ошибок.

Поиск по различным критериям. При проверке работы поиска, который поддерживает различные критерии (например, категория, дата, ключевые слова), важно убедиться, что результаты корректно соответствуют каждому выбранному критерию. Проверяем, что приложение правильно интерпретирует запросы пользователя и фильтрует данные в соответствии с указанными параметрами. Например, при поиске по категории приложение должно отображать только те результаты, которые относятся к выбранной категории, при поиске по дате – соответствующие временные промежутки, а при поиске по ключевым словам – результаты, содержащие эти слова.

Отсутствие результатов поиска. При тестировании реакции приложения на запросы, которые не возвращают результатов, важно убедиться, что пользователь получает адекватную и полезную обратную связь. Вместо пустой страницы или сообщения «ничего не найдено» приложение должно предлагать рекомендации по улучшению запроса. Это могут быть советы по уточнению ключевых слов, предложенные альтернативные поисковые запросы или возможность расширить критерии поиска.

⁷ Индикатор загрузки – это визуальный элемент интерфейса, который показывает, что процесс загрузки данных или выполнения задачи в приложении все ещё продолжается. Он может принимать различные формы, например, круговое вращение, прогресс-бар, анимированные изображения или текстовые подсказки «Загрузка...» и т. д.

Точность результатов поиска. Анализируя точность результатов поиска, важно убедиться, что приложение корректно отображает результаты, соответствующие введённому запросу. Это означает, что поиск должен возвращать релевантные данные как для точных совпадений, так и для частичных совпадений. Тестирование должно охватывать различные типы запросов, включая те, где запрос полностью совпадает с результатами, содержит частичное совпадение, опечатки или синонимы. Например, если пользователь вводит слово с ошибкой, приложение должно предложить корректные варианты или попытаться найти результаты, которые соответствуют предполагаемому значению запроса.

Читаемость результатов поиска. При анализе результатов поиска важно убедиться, что они представлены в ясной и понятной форме, обеспечивая пользователя достаточной информацией для оценки их соответствия запросу. Результаты должны включать ключевые детали, такие как заголовки, краткое описание, дату или другие релевантные сведения, чтобы пользователь мог быстро определить, подходит ли найденный результат, без необходимости переходить по ссылке. Это повышает удобство использования и позволяет пользователю более эффективно взаимодействовать с приложением.

Навигация по результатам. Проверая навигацию по результатам поиска, необходимо убедиться, что пользователю предоставлена удобная возможность перемещения по списку найденных элементов. Убеждаемся в наличии чётко обозначенных кнопок или ссылок для перехода между страницами с результатами, а также возможности прокрутки, если результаты представлены на одной странице. Прокрутка должна работать плавно, без задержек, обеспечивая доступ ко всем результатам поиска. Навигация должна быть интуитивной, позволяя пользователю легко находить и просматривать требуемую информацию, независимо от объёма данных.

Сортировка результатов. При тестировании функции сортировки результатов убеждаемся, что эта опция работает корректно и предсказуемо. Это включает в себя проверку того, что результаты поиска правильно сортируются в соответствии с выбранными пользователем критериями, такими как дата, релевантность, цена или алфавитный порядок. Необходимо протестировать различные варианты сортировки, чтобы убедиться, что каждый из них выполняет свою задачу и результаты отображаются в правильном порядке.

Поговорим о фильтрах. В данном контексте фильтры – это инструменты в приложениях, которые помогают уточнить или сузить результаты поиска по определённым критериям, чтобы пользователь мог быстрее найти нужную информацию. Например, при поиске книги в интернет-магазине можно использовать фильтры, чтобы отобразить книги только определённого жанра, автора или выпущенные в определённый период. Это удобный способ исключать лишние данные.

Уточнение результатов поиска и фильтрация. Пользователи должны иметь возможность уточнять поиск с помощью фильтров, категорий или изменения запроса. Проверяем, что пользователи могут сужать выборку данных с помощью доступных фильтров. Проверяем работу фильтров, чтобы они корректно применялись и отсеивали только те результаты, которые не соответствуют выбранным критериям. Необходимо также убедиться, что результаты после применения фильтров точно соответствуют заданным условиям.

Комбинирование фильтров. Изучая работу фильтров важно убедиться, что фильтры корректно работают в сочетании друг с другом, обеспечивая точное и ожидаемое сужение результатов поиска. Это означает, что при применении нескольких фильтров одновременно система должна учитывать все выбранные критерии и отображать только те результаты, которые соответствуют всем установленным условиям. Необходимо протестировать различные комбинации фильтров, чтобы убедиться, что они правильно взаимодействуют между собой и что результаты поиска точно отражают выбранные параметры. Например, если пользователь применяет фильтры по категории, цене и дате, отображаемые результаты должны удовлетворять всем этим критериям.

Сброс фильтров. Необходимо убедиться, что пользователь имеет возможность сбросить все применённые фильтры, чтобы вернуться к отображению всего списка результатов. Функция сброса должна быть легко доступной и интуитивно понятной, позволяя пользователю одним действием отменить все ранее выбранные фильтры. После сброса фильтров приложение должно корректно обновить результаты поиска, отобразив все доступные элементы без ограничений.

Интуитивность применения фильтров. При тестировании фильтров в поиске важно проверить, что они легко находятся и интуитивно понятны для пользователя. Фильтры должны быть хорошо видимы на странице и иметь чёткие обозначения, которые ясно указывают, какие данные они фильтруют. Пользователь должен сразу понимать, как применить фильтр и какие результаты он ожидает увидеть после его использования. Надписи, иконки или описания рядом с фильтрами должны быть максимально понятными и не оставлять места для двусмысленных интерпретаций.

Скорость работы фильтров. Интерфейс должен быстро реагировать на применение фильтров, предоставляя немедленную обратную связь о процессе фильтрации. После выбора фильтров пользователь должен сразу увидеть, что фильтрация выполняется, например, через появление индикатора загрузки или мгновенное обновление списка результатов. Даже при обработке больших объёмов данных фильтрация не должна заметно замедлять отображение результатов. Важно, чтобы фильтры работали плавно и без задержек.

Интеграция фильтров с другими компонентами. При проверке интеграции фильтров с другими компонентами приложения, такими как пагинация⁸ или сортировка результатов поиска, требуется убедиться, что все элементы взаимодействуют корректно. Применение фильтров не должно нарушать работу пагинации – при фильтрации результаты должны правильно разбиваться на страницы, и пользователь должен иметь возможность переходить между страницами отфильтрованных данных. Также необходимо проверить, что фильтры корректно работают в сочетании с функцией сортировки и результаты после применения фильтров остаются отсортированными в соответствии с выбранными критериями. Эти проверки гарантируют, что все компоненты приложения работают согласованно.

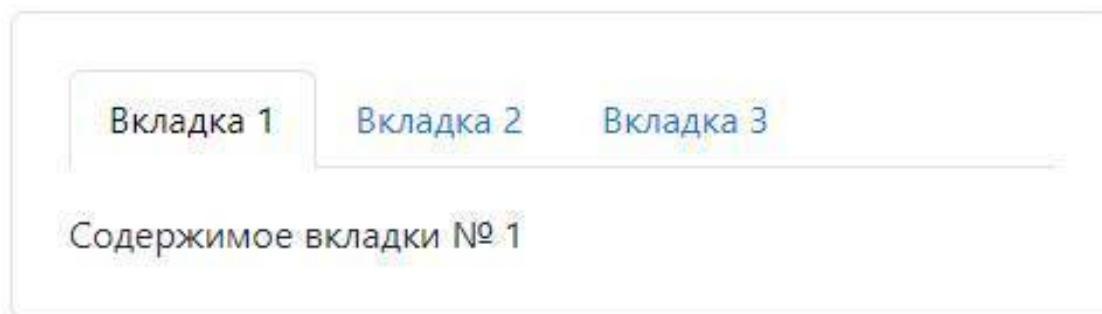
Граничные значения и нестандартный ввод данных. При тестировании фильтров необходимо убедиться, что они правильно обрабатывают как минимальные, так и максимальные допустимые значения. Например, если фильтр предназначен для отбора товаров по цене, нужно проверить, что фильтрация корректно работает при установке предельных значений (минимально возможной и максимально возможной цены). Также следует протестировать фильтры с нестандартным вводом данных, такими как отрицательные числа, нули или очень большие значения, чтобы убедиться, что система правильно обрабатывает такие случаи. Это может включать как правильную фильтрацию, так и вывод предупреждений или сообщений об ошибках, если введённые данные не подходят.

Динамическое обновление данных. Если данные, подлежащие фильтрации, могут динамически обновляться (в реальном времени) необходимо убедиться, что фильтры корректно реагируют на изменения данных. Это означает, что если в приложении данные обновляются или добавляются динамически (например, новые товары на складе или изменения в списке доступных услуг), результаты поиска должны мгновенно отражать эти изменения при включённых фильтрах. Пользователь должен видеть актуальные результаты, соответствующие выбранным критериям, без необходимости вручную обновлять страницу или повторно применять фильтры.

⁸ Пагинация – это способ разделения содержимого в приложении на несколько страниц, чтобы упростить навигацию и просмотр. Проще говоря, вместо того чтобы сразу загружать всю информацию, которая может быть очень большой, пагинация позволяет отображать её частями постранично.

Тестирование вкладок

Вкладки позволяют организовать содержимое в логически разделённые секции, доступные для просмотра без перезагрузки страницы.



Рассмотрим основные аспекты, требующие внимания при проверке вкладок.

Переключение между вкладками. При тестировании вкладок необходимо убедиться, что переключение между ними происходит корректно и отображается соответствующее содержимое для каждой активной вкладки. Переключение должно происходить без перезагрузки всей страницы или формы приложения. Каждая вкладка должна отображать именно то содержимое, которое соответствует её названию, и это содержимое должно быть доступно сразу после переключения, без задержек или ошибок.

Состояние активной вкладки. Активная вкладка всегда должна чётко выделяться визуально среди других. Это может быть достигнуто с помощью цветового выделения, изменения шрифта, подчёркивания или других визуальных эффектов, чтобы пользователь сразу мог определить, какая вкладка активна. Кроме того, при переключении между вкладками необходимо проверить, что состояние активности корректно обновляется – предыдущая вкладка теряет выделение, а новая вкладка становится активной и выделяется соответствующим образом.

Выбор по умолчанию. При первом открытии страницы или формы приложения автоматически должна активироваться определённая вкладка, как это предусмотрено требованиями. Обычно это первая вкладка, но могут быть исключения в зависимости от специфики приложения. Сразу после загрузки страницы должно отображаться содержимое этой активной по умолчанию вкладки без необходимости дополнительных действий со стороны пользователя.

Сохранение состояния. Проверяем, что при перезагрузке страницы или возвращении на неё активная вкладка сохраняет своё состояние и остаётся активной, если это предусмотрено логикой работы приложения. Это означает, что если пользователь переключился на определённую вкладку и затем перезагрузил страницу, то после перезагрузки именно эта вкладка должна автоматически открыться снова вместе с соответствующим ей содержимым.

Фокус. При переключении на новую вкладку фокус должен перемещаться на связанное с ней содержимое или на саму вкладку. Это помогает пользователям, использующим клавиатуру или вспомогательные технологии, быстро ориентироваться в приложении. После переключения фокус может быть установлен либо на определённый элемент содержимого вкладки, либо на саму вкладку, что позволит пользователю сразу начать взаимодействовать с нужной частью приложения.

Быстродействие. Необходимо убедиться, что переключение между вкладками происходит быстро и без заметных задержек, даже если содержимое вкладок объёмное. Пользователь не должен испытывать задержек при переходе от одной вкладки к другой, так как это может снизить эффективность работы с приложением. Тестирование должно включать сценарии с

большим объёмом данных или сложными элементами внутри вкладок, чтобы убедиться, что приложение справляется с задачей без снижения производительности.

Взаимодействие с URL. При переключении между вкладками URL страницы должен корректно обновляться, если это предусмотрено функциональностью приложения. К примеру, URL может содержать якоря (например, #вкладка1) или параметры (например, ?tab=2), что позволяет пользователю сохранить или отправить ссылку, ведущую непосредственно к определённой вкладке. Это также полезно для улучшения навигации, поскольку пользователь может вернуться на нужную вкладку или поделиться ссылкой на конкретное содержимое. Тестирование должно включать проверку того, что URL корректно обновляется при переключении между вкладками и что при переходе по обновлённому URL открывается соответствующая вкладка с правильным содержимым.

<https://victorz.ru/book/book-2?tab=1>

Вкладка 1

Вкладка 2

Вкладка 3

Содержимое вкладки № 1

<https://victorz.ru/book/book-2?tab=2>

Вкладка 1

Вкладка 2

Вкладка 3

Содержимое вкладки № 2

Взаимодействие с другими элементами. Оцениваем, как переключение между вкладками влияет на другие элементы интерфейса, особенно если содержимое вкладки содержит в себе интерактивные элементы или формы. Необходимо убедиться, что при переходе на другую вкладку интерактивные элементы, такие как кнопки, поля ввода или другие формы, работают корректно и сохраняют своё состояние, если это предусмотрено логикой приложения. Важно проверить, что данные, введённые в форму на одной вкладке, не теряются при переключении на другую вкладку и обратно и что все элементы продолжают функционировать без ошибок.

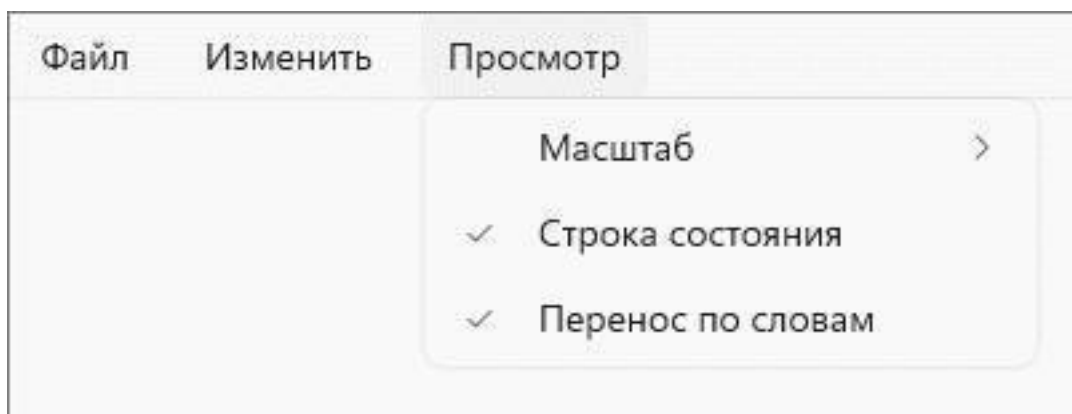
Изменение внешнего вида при наведении. Необходимо убедиться, что стиль вкладок изменяется при наведении на них курсора мыши, чётко указывая на их интерактивность. Это может включать изменение цвета, добавление подчёркивания, изменение шрифта или смену

курсора на указатель руки. Эти визуальные изменения помогают пользователю понять, что вкладка является активным элементом интерфейса, с которым можно взаимодействовать.

Тестирование меню приложений

Меню приложения – это список доступных функций или опций, который обычно располагается в верхней части окна приложения или в отдельной панели. Меню может включать в себя выпадающие списки, контекстные меню, меню на панели инструментов и другие элементы интерфейса, позволяющие пользователям взаимодействовать с приложением. В меню можно найти команды для открытия файлов, настройки параметров, выполнения операций и другие возможности.

На картинке видим меню, имеющее пункты «Файл», «Изменить», «Просмотр» с вложенными пунктами меню (подменю):



Выделим и обсудим основные аспекты, на которые стоит обратить внимание в процессе тестирования меню приложения.

Доступность функций. При тестировании меню приложения важно убедиться, что все заявленные функции доступны через меню и работают корректно. Убеждаемся, что все пункты меню ведут к нужным разделам или функциям приложения, как это предусмотрено спецификацией. Пользователь должен иметь возможность находить и использовать все основные и дополнительные функции через меню.

Вложенные меню. При проверке меню с вложенными подменю важно убедиться, что они корректно раскрываются и остаются полностью работоспособными. Проверяем, что при наведении курсора или клике на пункт меню соответствующее подменю плавно и без задержек открывается, отображая все свои элементы. Также следует убедиться, что все элементы внутри подменю активны, вызывают требуемые функциям или ведут к требуемым разделам приложения.

Горячие клавиши. Если для активации действий пунктов меню предусмотрены горячие клавиши, важно убедиться в их корректной работе и отсутствии конфликтов с другими командами. Убеждаемся, что нажатие соответствующих клавиш действительно приводит к выполнению ожидаемых действий, связанных с пунктами меню. Также следует убедиться, что горячие клавиши не конфликтуют с другими командами или системными сочетаниями клавиш, чтобы избежать нежелательных действий или путаницы. Тестирование должно охватывать все заявленные горячие клавиши, включая различные комбинации, чтобы гарантировать их надёжную работу и удобство использования для пользователя.

Логика расположения. Проверяем, что пункты меню организованы логически и интуитивно понятно, чтобы пользователи могли без труда находить нужные функции. Логика расположения должна отражать частоту использования и взаимосвязь между функциями – наиболее важные и часто используемые элементы должны быть легко доступны и находиться на

видных местах. Пункты меню группируются по смыслу или категории, чтобы пользователи сразу понимали, где искать нужную опцию.

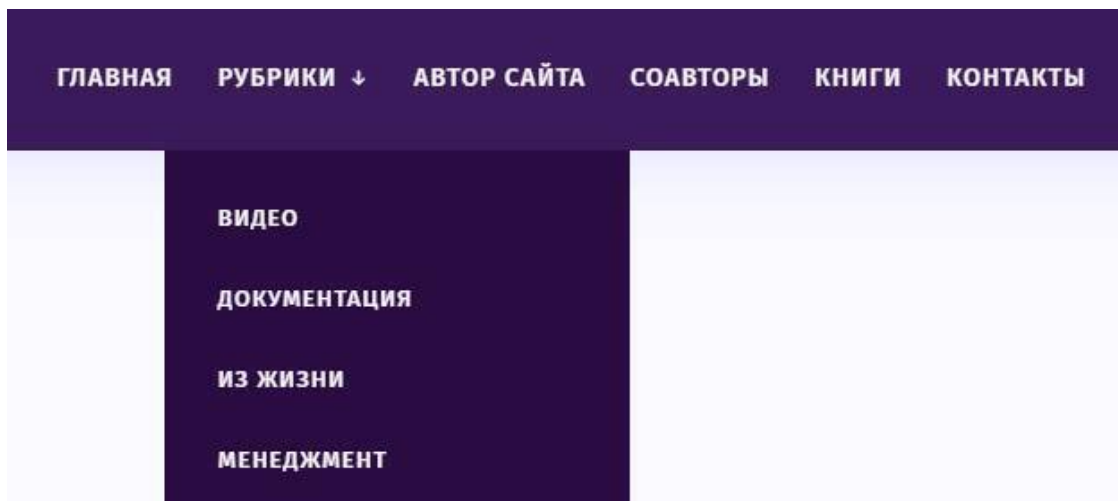
Отзывчивость. Необходимо убедиться, что меню работает быстро и без задержек, обеспечивая комфортное взаимодействие для пользователя. Это означает, что при клике или наведении курсора меню должно мгновенно открываться, предоставляя доступ к своим пунктам, и также быстро закрываться, когда пользователь завершил взаимодействие.

Адаптация под условия использования. Меню должно адаптироваться к различным условиям использования. Это означает, что необходимо проверить, если применимо, чтобы меню изменяло состав пунктов меню в зависимости от текущего контекста, активных модулей или задач, выполняемых пользователем. Например, меню может отображать разные наборы опций в зависимости от того, какой раздел приложения открыт или какие действия недавно были выполнены.

Изменение внешнего вида при наведении. Проверяем, что стиль пунктов меню изменяется при наведении на них курсора мыши, чётко указывая на их интерактивность. Это может включать изменение цвета фона или текста, добавление подчёркивания, увеличение контраста или смену курсора на указатель руки. Эти визуальные изменения помогают пользователю понять, что на данный пункт меню можно кликнуть, и делают навигацию по меню интуитивной и удобной.

Тестирование меню навигации сайтов

Меню навигации сайта – это набор ссылок или кнопок, которые помогают пользователям перемещаться по различным разделам и страницам сайта. Оно обычно располагается в верхней части страницы или по бокам и содержит основные категории или разделы сайта, чтобы пользователи могли быстро и удобно найти нужную информацию или перейти на нужную страницу.



Тестирование меню навигации сайтов важно для обеспечения удобства использования сайта и его доступности для пользователей. Давайте изучим ключевые моменты, которые необходимо учесть в процессе тестирования меню навигации.

Переходы по ссылкам. При тестировании меню сайтов важно убедиться, что все ссылки в меню навигации корректно работают и ведут на целевые страницы без ошибок. Проверяем, что каждая ссылка в меню правильно направляет пользователя на нужный раздел сайта и что при переходе по ссылке не возникает ошибок, таких как 404⁹. Необходимо протестировать все ссылки, чтобы исключить наличие «мёртвых» ссылок, которые никуда не направляют пользователя и негативно влияют на общее впечатление от сайта.

Вложенные пункты меню. Проверяя меню, содержащее вложенные подменю или выпадающие списки, важно убедиться в его корректном отображении и функциональности. Проверяем, что подменю правильно раскрываются при наведении курсора или клике, отображая все вложенные пункты без задержек и ошибок. Необходимо также убедиться, что все элементы внутри подменю интерактивны и ведут на правильные страницы или выполняют нужные функции. Вложенные меню должны работать плавно и без ошибок, не перекрывая другие элементы интерфейса и обеспечивая удобную навигацию по сайту.

Интерактивные элементы. При анализе работы меню, которое содержит интерактивные элементы, такие как кнопки или формы поиска, важно убедиться в их корректной работе. Подтверждаем, что кнопки выполняют свои функции, например, открывают подменю, отправляют данные или выполняют другие действия, и что формы поиска правильно принимают и обрабатывают пользовательские запросы. Также следует удостовериться, что эти элементы не конфликтуют с другими частями меню и интерфейса, работают плавно и предоставляют пользова-

⁹ Ошибка 404 – страница или запрашиваемый ресурс не найдены. Рекомендую дополнительно изучить информацию о кодах состояния HTTP (коды ответа сервера).

телю ожидаемую обратную связь. Например, кнопка должна визуально реагировать на нажатие, а форма поиска должна отображать результаты без ошибок.

Визуальное выделение активного пункта меню. Необходимо убедиться, что активный пункт меню, соответствующий текущему разделу или странице, на которой находится пользователь, визуально выделен. Это выделение может включать изменение цвета фона или текста, добавление подчёркивания или изменение стиля шрифта. Визуальное выделение помогает пользователю легко определить, в каком разделе сайта он находится, что делает интерфейс интуитивно понятным. Данная функция особенно важна для ориентации на сайте, так как позволяет пользователю всегда видеть, где он находится в структуре меню, и легко переходить между различными разделами.

Меню на разных страницах. При тестировании необходимо убедиться, что меню сохраняет свою функциональность и визуальное отображение на всех страницах, независимо от их содержания. Это означает, что меню должно правильно отображаться и работать как на главной странице, так и на внутренних страницах, где может присутствовать различное количество текста, изображений или других элементов.

Тестирование изображений

Тестирование иконок и изображений в приложениях крайне важно для обеспечения качественного визуального восприятия пользователями приложения. Опишем ключевые аспекты, на которые следует обратить внимание.

Разрешение и чёткость. При тестировании изображений важно убедиться, что они сохраняют чёткость и высокое качество на всех устройствах и при разных разрешениях экрана. Это означает, что изображения должны выглядеть одинаково хорошо как на больших мониторах, так и на мобильных устройствах, без пикселизации или размытия. Особое внимание следует уделить их визуальному качеству при масштабировании – изображения должны оставаться чёткими и не терять детализацию при увеличении или уменьшении размера.

Цвет и контраст. При проверке изображений необходимо убедиться, что их цвета обеспечивают достаточный контраст с фоновыми элементами для лёгкого восприятия всеми пользователями, включая людей с ограниченными возможностями зрения. Изображения должны быть легко различимы и не сливаться с фоном, что особенно важно для пользователей с нарушением цветового восприятия или слабовидящих. Использование цветовых схем с достаточным контрастом не только улучшает общую визуальную доступность, но и делает интерфейс более удобным и понятным для всех категорий пользователей.

Взаимодействие с пользователем. Необходимо убедиться, что иконки и изображения, которые предполагают взаимодействие с пользователем (например, кнопки соцсетей, иконки для скачивания), корректно работают. Проверяем, что при нажатии или клике на такие иконки происходит ожидаемое действие: переход на страницу социальной сети, скачивание файла или выполнение другой запланированной функции. Необходимо также проверить, что эти элементы визуально сигнализируют о своей интерактивности, например, изменением цвета, появлением анимации или изменением курсора при наведении.

Всплывающие подсказки. У иконок, не имеющих явного текстового описания, важно проверить наличие всплывающих подсказок, которые помогают пользователю понять их функции. Всплывающие подсказки должны появляться при наведении курсора на иконку и содержать краткое, но информативное описание действия или назначения иконки.

Время загрузки. Необходимо убедиться, что изображения оптимизированы для быстрой загрузки, особенно на мобильных устройствах с ограниченной пропускной способностью интернета. Изображения должны загружаться быстро и плавно, не замедляя работу сайта и не ухудшая пользовательский опыт.

Формат и сжатие. При тестировании изображений важно убедиться, что используются подходящие форматы файлов и правильные методы сжатия, чтобы минимизировать размер файлов без значительной потери качества. Например, фотографии следует сохранять в формате JPEG, который обеспечивает хорошее качество при относительно небольшом размере файла, а иконки с прозрачностью – в формате PNG, который поддерживает прозрачность и сохраняет резкость изображений. Важно также проверить, что файлы эффективно сжаты, чтобы ускорить их загрузку, но при этом не потерять визуальное качество до такой степени, что это станет заметно пользователю. Оптимизация графики особенно критична для мобильных пользователей, где скорость загрузки может значительно повлиять на восприятие.

Альтернативный текст. При проверке изображений на сайте важно убедиться, что для каждого из них предусмотрен альтернативный текст (атрибут «alt»), который точно и корректно описывает содержимое изображения. Альтернативный текст играет ключевую роль в обеспечении доступности сайта для пользователей, использующих скринридеры¹⁰, а также для

¹⁰ Скринридер – это программа, которая помогает людям с нарушениями зрения использовать компьютеры и мобильные

тех, у кого изображения не загружаются. Этот текст должен быть кратким, но достаточно информативным, чтобы передать основное содержание или функцию изображения. Например, вместо общего описания «картинка» должно использоваться конкретное описание, такое как «логотип компании» или «фото продукта».

Интеграция с содержимым форм и страниц. При анализе работы интеграции изображений и иконок с содержимым форм и страниц приложений важно убедиться, что они гармонично вписываются в общий дизайн и не создают проблем для восприятия информации. Проверяем, что изображения и иконки не перекрывают текст, поля ввода или другие важные элементы интерфейса. Также необходимо убедиться, что их размещение и размер не нарушают структуру страницы – они должны естественно дополнять содержание, а не отвлекать или создавать визуальный беспорядок.

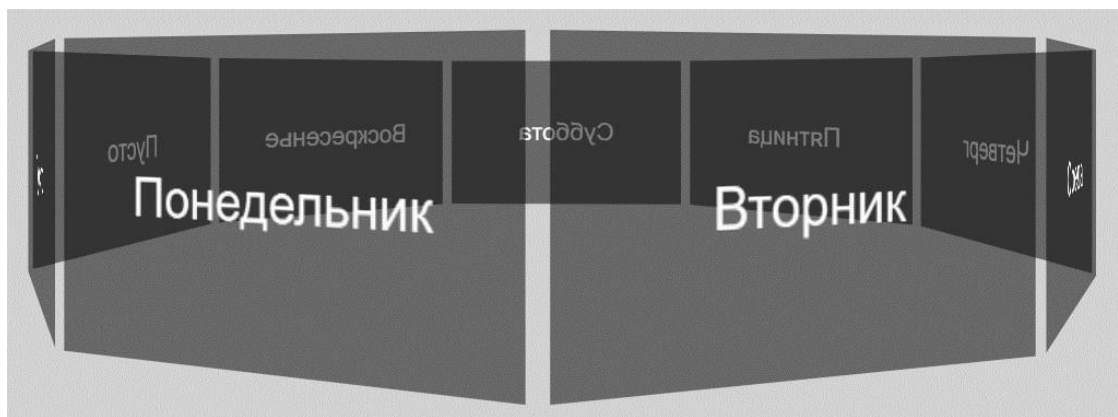
Тестирование слайдеров и каруселей

Слайдер и карусель – это элементы интерфейса в приложениях, которые позволяют просматривать ряд изображений, текстов или другого содержимого, перелистывая их вперёд или назад.

Слайдер обычно представляет собой горизонтальную полосу, по которой можно пролистывать содержимое влево или вправо. Чаще всего слайдеры используются для показа изображений: например, фотографий товаров в интернет-магазине. Слайдер может автоматически менять изображения через определённые интервалы времени или позволять пользователю перелистывать их вручную с помощью кнопок или жестов:



Карусель похожа на слайдер, но часто подразумевает возможность показа нескольких элементов одновременно. Например, в карусели могут отображаться несколько фотографий товаров рядом, и при перелистывании они все вместе смещаются влево или вправо, открывая место для следующей группы элементов. Карусели используются для экономии места на экране, позволяя разместить больше содержимого в ограниченной области видимости:



Оба этих элемента удобны для представления визуального содержимого и могут включать различные функции, такие как автопроигрывание, анимация перелистывания, индикаторы текущего положения и кнопки для навигации. Тестирование слайдеров и каруселей в интерфейсах приложений касается проверки их функциональности, визуального представления, удобства использования и совместимости. Рассмотрим, на что стоит обратить внимание при тестировании.

Плавное перелистывание. Необходимо убедиться, что перелистывание слайдов или элементов карусели происходит плавно и без задержек. Это означает, что при смене слайда или элемента карусели анимация должна быть плавной, без рывков или зависаний. Задержки или прерывания в анимации могут негативно повлиять на восприятие интерфейса и снизить удобство использования.

Управление. Проверяем наличие и корректную работу элементов управления, таких как «вперёд», «назад», «смена слайда». Эти элементы должны чётко выполнять свою функцию, позволяя пользователю легко переключаться между слайдами или элементами карусели. Кроме того, необходимо удостовериться, что на сенсорных устройствах предусмотрена возможность перелистывания слайдов с помощью жестов, таких как свайп¹¹. Эти жесты должны работать без ошибок, обеспечивая плавный переход между слайдами.

Автопроигрывание. Если слайдер или карусель имеет функцию автоматического проигрывания (пролистывания), проверяем, что эта функция работает корректно. Проверяем, что слайды или элементы карусели автоматически пролистываются с заданным интервалом времени и делают это плавно и без задержек. Также необходимо удостовериться, что пользователи имеют возможность приостановить автопроигрывание, например, через элемент управления «пауза» или путём взаимодействия со слайдером (наведение курсора, свайп).

Время загрузки. При тестировании слайдеров и каруселей важно убедиться, что они не замедляют загрузку страницы, особенно при её первом открытии пользователем. Слайдеры и карусели часто содержат множество изображений, анимаций и скриптов, что может увеличить время загрузки из-за большого объёма данных и необходимости выполнения дополнительных операций. Для обеспечения быстрого открытия страниц следует оптимизировать эти элементы: использовать сжатые и оптимизированные изображения, внедрять техники отложенной загрузки для контента слайдера, минимизировать и объединять скрипты и стили.

Корректная обработка ошибок. Необходимо убедиться, что слайдер или карусель корректно обрабатывают ошибки, такие как невозможность загрузить изображение или другой контент. В случае, если изображение не загружается, слайдер или карусель должны оставаться структурно целыми, без искажения своей формы или внешнего вида. Например, вместо не

¹¹ Свайп – управляющий жест при работе с сенсорным экраном, при котором палец кладут на экран и проводят в каком-либо направлении.

загруженного изображения может отображаться запасной элемент, такой как иконка или сообщение о том, что изображение недоступно. Также важно проверить, что отсутствие одного элемента не нарушает общую структуру страницы, на которой расположен слайдер или карусель.

Тестирование прогресс-баров

Тестирование прогресс-баров в приложениях важно для обеспечения точного информирования пользователей о статусе выполнения длительных процессов, таких как загрузка файлов, выполнение вычислений или обработка данных. Пример:



Обратим внимание на важные детали, касающиеся проверки прогресс-баров.

Точность отображения. Проверяем, что прогресс-бар точно отображает прогресс выполнения задачи. Это означает, что визуальное представление прогресса должно соответствовать реальному состоянию задачи. Например, если задача завершена на 80%, прогресс-бар должен чётко показывать 80% заполнения, без отклонений или ошибок в отображении. Точность прогресс-бара особенно важна для обеспечения прозрачности в процессе выполнения задач, так как любые несоответствия могут вызвать путаницу.

Обновление в реальном времени. При тестировании прогресс-бара важно убедиться, что он обновляется в реальном времени, предоставляя пользователю актуальную информацию о ходе выполнения задачи. Это означает, что по мере продвижения задачи прогресс-бар должен динамически отражать изменения, без задержек или необходимости обновления страницы. Обновление в реальном времени помогает пользователю оценивать, сколько времени осталось до завершения задачи.

Соответствие скорости выполнения. При анализе работы прогресс-бара важно убедиться, что скорость изменения его состояния соответствует фактической скорости выполнения процесса. Прогресс-бар должен обновляться плавно и равномерно, без внезапных скачков или длительных застоев, если сам процесс выполняется равномерно. Это означает, что пользователю должна предоставляться точная визуальная индикация того, как быстро или медленно движется задача. Любые несоответствия, такие как резкие скачки заполнения или задержки при равномерном процессе, могут создать у пользователя ложное представление о ходе выполнения задачи.

Отклик на ошибки. Проверяем, что прогресс-бар корректно реагирует на ошибки и прерывания процесса. Если процесс, такой как загрузка файла, прерывается из-за ошибки, прогресс-бар должен сбросить своё состояние в начальное или соответствующим образом отобразить, что задача не завершена. Кроме того, приложение должно незамедлительно предупредить пользователя о возникшей проблеме, предоставив чёткое и понятное сообщение об ошибке. Это сообщение должно информировать пользователя о том, что произошло, и по возможности предложить дальнейшие шаги для решения проблемы.

Приостановка и продолжение процесса. При тестировании возможностью приостановки или отмены процесса пользователем необходимо убедиться, что прогресс-бар корректно реагирует на эти действия. Если пользователь решает приостановить процесс, прогресс-бар должен зафиксировать своё текущее состояние и остановиться, визуально показывая, что процесс временно прерван. В случае отмены процесса прогресс-бар должен сброситься в начальное состояние, полностью очистившись или скрывшись, в зависимости от логики работы приложения. Кроме того, приложение должно предоставить пользователю чёткую обратную связь, подтверждающую, что процесс был приостановлен или отменён.

Обработка больших файлов или данных. Особое внимание уделяем проверке работы прогресс-баров при работе с большими файлами или объёмами данных, чтобы убедиться в их точности и надёжности. В таких случаях прогресс-бар должен точно отображать прогресс выполнения задачи на протяжении всего процесса, независимо от его продолжительности и объёма обрабатываемых данных. Это включает проверку того, что прогресс-бар не зависает, не сбивается и не отображает некорректный прогресс, даже если процесс занимает много времени.

Тестирование видео- и аудиоплееров

Тестирование видео- и аудиоплееров в приложениях крайне важно для обеспечения качественного воспроизведения мультимедийного содержимого и удобства его использования пользователями. Рассмотрим ключевые аспекты, на которые следует обратить внимание.

Воспроизведение является ключевым аспектом работы видео- и аудиоплееров, и при тестировании нужно убедиться, что оно происходит плавно, без каких-либо задержек, зависаний или прерываний. Это означает, что и видео, и аудио должны воспроизводиться синхронно, без «заиканий», дефектов или рассинхронизации между звуком и изображением. Для точного тестирования важно проводить его в условиях максимально стабильного и быстрого интернет-соединения. Это позволяет исключить влияние проблем с сетью и сфокусироваться на выявлении возможных проблем непосредственно в плеере. Важно, чтобы соединение между сервером, где хранятся медиафайлы, и устройством, на котором проводится тестирование, было максимально быстрым и надёжным. Таким образом, если в процессе тестирования возникают проблемы с воспроизведением, можно с уверенностью утверждать, что они связаны именно с работой плеера, а не с качеством интернет-соединения.

Элементы управления. Проверка всех элементов управления с целью убедиться в их корректной работе. В первую очередь проверяются основные кнопки, такие как воспроизведение и пауза, чтобы убедиться, что плеер запускает и останавливает воспроизведение медиафайлов. Далее проводится тестирование регулировки громкости, при котором проверяется, как отображаются и влияют на воспроизведение изменения уровня звука. Тестируются функции перемотки вперёд и назад для оценки их точности и отклика. Проверяется переключение качества воспроизведения, чтобы убедиться, что плеер правильно адаптируется к выбранным настройкам, например, изменяет разрешение видео. Также тестируется полноэкранный режим, чтобы убедиться, что переход в полноэкранный формат и выход из него работают без задержек и артефактов. В ходе тестирования обращается внимание на стабильность плеера, его реакцию на пользовательские команды и отсутствие ошибок.

Отображение прогресса. Тестируется корректность работы прогресс-бара, который показывает текущую позицию воспроизведения медиафайла. Важно убедиться, что индикатор прогресса точно отражает реальное время, отображая движение по шкале в соответствии с фактическим воспроизведением аудио или видео. Прогресс-бар должен обновляться плавно и без задержек, синхронизируясь с воспроизведением. Кроме того, пользователь должен иметь возможность изменять текущую позицию воспроизведения, перетаскивая индикатор по шкале или кликая на нужную точку, без сбоя или задержек. При этом плеер должен мгновенно реагировать на изменения, корректно продолжая воспроизведение с выбранного момента.

Адаптивное воспроизведение. Адаптивное воспроизведение предполагает автоматическую подстройку качества видео под текущую скорость интернет-соединения, чтобы обеспечить максимально плавное и стабильное воспроизведение. Тестирование этой функции включает проверку работы плеера в условиях различных сетевых сценариев: от высокоскоростных до медленных и нестабильных соединений. Во время тестирования важно убедиться, что плеер корректно меняет качество видео, например, снижает разрешение при падении скорости интернета и повышает его, когда соединение становится лучше. Также проверяется, чтобы процесс изменения качества происходил плавно, без прерываний или заметных задержек. Кроме того, тестируется, насколько быстро и точно плеер реагирует на изменения в состоянии сети, обеспечивая оптимальное качество изображения в каждом конкретном случае.

Качество аудио и видео. При проверке качества аудио и видео в плеере важно убедиться, что пользователю доступны различные настройки качества и что они применяются корректно. Тестирование начинается с проверки наличия опций для изменения качества, таких как выбор

разрешения видео (например, 480p, 720p, 1080p) и настройки битрейта аудио. Далее необходимо проверить, что при выборе различных параметров качество воспроизведения действительно изменяется: изображение становится более чётким или менее детализированным, звук – лучше или хуже в зависимости от выбранных настроек. Особое внимание уделяется плавности перехода между разными режимами качества, чтобы не возникало задержек, прерываний или артефактов при переключении. Также проверяется, что настройки сохраняются и применяются сразу после выбора, обеспечивая пользователю предсказуемый и стабильный опыт при изменении параметров воспроизведения.

Доступность. Плеер должен быть доступен для пользователей с ограниченными возможностями. В ходе тестирования оценивается, могут ли все основные функции плеера, такие как воспроизведение, пауза, регулировка громкости, перемотка и переключение качеств, управляться с помощью клавиатуры. Проверяется наличие и корректная работа клавиатурных комбинаций, позволяющих быстро выполнять действия без использования мыши, таких как пробел для паузы, стрелки для перемотки и другие горячие клавиши.

Различные типы воспроизводимых данных. Анализируем работу плеера на различных типах воспроизводимых данных с целью проверки его способности корректно работать с разными форматами аудио- и видеофайлов. В процессе тестирования используются наиболее распространённые форматы видео, такие как MP4, AVI, MKV, MOV, и аудиоформаты, включая MP3, WAV, AAC, FLAC. Проверяется, что плеер безошибочно распознает и воспроизводит каждый из этих форматов, обеспечивая плавное и стабильное воспроизведение. Также важно убедиться, что плеер справляется с различными кодеками, используемыми в этих форматах, и может корректно обрабатывать различные разрешения и битрейты. Тестирование проводится на разных типах файлов, чтобы выявить возможные проблемы с несовместимостью или нестабильностью, подтверждая, что плеер способен воспроизводить широкий спектр медиафайлов без потери качества или функциональности.

Тестирование модальных окон

Модальные окна являются важным компонентом интерфейса приложений, предоставляя пользователю важную информацию или запрашивая подтверждение действий без перезагрузки текущей страницы или изменения контекста работы.



При тестировании модальных окон важно обратить внимание на ряд аспектов.

Открытие и закрытие. В ходе тестирования проверяется, что модальное окно открывается при выполнении соответствующих действий, таких как нажатие на кнопку или ссылку, и отображается правильно, без визуальных дефектов. Далее проводится проверка всех доступных способов закрытия окна: нажатие на кнопку закрытия (например, крестик в углу), использование специальных кнопок внутри окна (таких как «Отмена» или «Сохранить и закрыть»), а также клики вне области модального окна, если такая возможность предусмотрена. Важно удостовериться, что закрытие происходит плавно и без задержек, а модальное окно больше не мешает взаимодействию с основным интерфейсом после завершения действия.

Закрытие с клавиатуры. Пользователи должны иметь возможность закрыть модальное окно, используя клавиатуру (например, нажатием клавиши **Escape**).

Фокусировка. Проверяем, что при открытии модального окна фокус автоматически перемещается на первый интерактивный элемент внутри окна, обеспечивая доступность для пользователей, работающих с клавиатурой. Это необходимо для того, чтобы пользователь сразу мог начать взаимодействие с модальным окном, не теряя время на ручное перемещение фокуса. Важно убедиться, что при открытом модальном окне нажатия мыши и клавиатурные действия не влияют на элементы за его пределами. Проверяется, что фокус и клавиатурная навигация полностью ограничены рамками модального окна, и элементы заднего фона остаются недоступными до его закрытия. После закрытия модального окна доступность и взаимодействие с элементами основного интерфейса должны восстанавливаться в полном объёме.

Задний фон. Проверяем, что элементы интерфейса за модальным окном становятся неактивными, блокируя взаимодействие пользователя с основной страницей до закрытия модального окна. Для этого фоновый контент обычно визуально отделяется от модального окна с помощью затемнения или размытия, что подчёркивает активность модального окна и предотвращает случайные клики по элементам заднего фона.

Элементы управления и ссылки. Анализируя работу элементов управления и ссылок внутри модального окна, проверяется их корректная работа и соответствие ожидаемым функциям. Для этого проводится проверка всех интерактивных элементов, включая кнопки, ссылки, чекбоксы, поля ввода и выпадающие списки. Каждая кнопка и ссылка должны выполнять свою функцию, например, кнопка «Закрыть» должна закрывать окно, а кнопка «Сохранить» – сохранять введённые данные и закрывать модальное окно или оставаться активной в

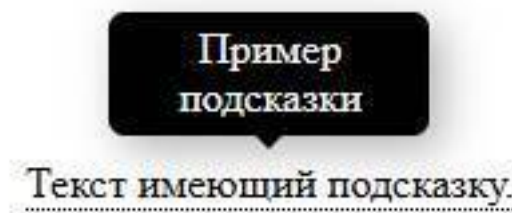
зависимости от логики. Также проверяются ссылки, чтобы они корректно перенаправляли на нужные страницы или выполняли соответствующие действия внутри окна.

Обработка ошибок. При тестировании окон, используемых для отображения сообщений об ошибках или предупреждений, необходимо убедиться, что все сообщения корректно отображаются и соответствуют ситуации, вызвавшей их появление. Проверяется, что окно открывается при возникновении ошибки и что текст сообщения полностью виден, легко читается и точно отражает суть проблемы. Важно проверить наличие соответствующих визуальных индикаторов, таких как значки предупреждений или ошибок.

Множество модальных окон. Когда возможно последовательное открытие нескольких окон, важно проверить их корректное управление и взаимодействие. Необходимо убедиться, что при открытии нового модального окна предыдущее окно становится неактивным, но остаётся видимым в фоновом режиме, если это предусмотрено логикой приложения. Проверяется, что фокус автоматически перемещается на активное окно и все элементы управления внутри него работают корректно. При закрытии активного окна фокус должен возвращаться на предыдущее, а его функциональность – восстанавливаться. Важно убедиться, что порядок их открытия и закрытия соответствует ожидаемому поведению. Тестируется также, что при попытке закрытия всех окон они закрываются последовательно в обратном порядке открытия, и интерфейс возвращается в исходное состояние, не оставляя неактивных или «зависших» окон.

Тестирование всплывающих подсказок

Всплывающие подсказки представляют собой информационные сообщения, которые появляются при наведении курсора мыши или при касании элемента на сенсорном экране, предоставляя дополнительную информацию о функции или элементе интерфейса.



Рассмотрим детали, на которые требуется обратить внимание при тестировании всплывающих подсказок.

Открытие и закрытие подсказок. При тестировании необходимо убедиться, что подсказки корректно активируются и появляются в нужный момент. Проверяется активация подсказок при наведении курсора мыши на соответствующий элемент, чтобы они отображались без задержек и чётко указывали на информацию, которую должны передать. Также тестируется активация подсказок при помощи клавиатуры: фокус должен перемещаться на элемент с подсказкой, а сама подсказка должна появляться автоматически или при выполнении соответствующего действия, например, нажатия клавиши «Enter» или пробела. На сенсорных устройствах проверяется, что подсказки появляются при касании соответствующего элемента экрана. Важно убедиться, что подсказки исчезают, когда пользователь убирает курсор, теряет фокус или отводит палец, и не остаются видимыми дольше, чем необходимо, чтобы не мешать взаимодействию с интерфейсом.

Точность и соответствие. Анализируя работу подсказок, проверяем, что их содержимое точно отражает функциональность или информацию, связанную с элементом, для которого они предназначены. Подсказки должны быть понятными, краткими и давать пользователю ясное представление о действии или значении элемента и соответствовать контексту: текст должен точно описывать назначение элемента интерфейса, рядом с которыми она появляется, избегая двусмысленностей или сложных формулировок. Важно убедиться, что в подсказках отсутствуют ошибки, такие как опечатки, неправильная информация или устаревшие данные, которые могут ввести пользователя в заблуждение.

Читаемость. Проверяем, что цвет текста и фона подобраны так, чтобы обеспечивать хорошую видимость даже для пользователей с нарушениями зрения, включая дальтонизм. Текст должен быть достаточно крупным и чётким, без мелкого шрифта, размытости или эффектов, затрудняющих чтение. Тестируются различные сценарии отображения, чтобы убедиться, что текст не сливается с фоном и не исчезает на светлых или темных участках экрана. Также важно проверить читаемость подсказок при разных настройках яркости экрана и в различных режимах, таких как ночной или тёмный режим, чтобы гарантировать их доступность и удобство в любых условиях использования.

Позиционирование. При тестировании позиционирования всплывающих подсказок необходимо убедиться, что они отображаются в правильном месте и не перекрывают важные элементы интерфейса, а также сам элемент, к которому они относятся. Подсказки должны располагаться так, чтобы не затруднять пользователю просмотр и взаимодействие с интерфейсом. Важно протестировать их поведение при масштабировании содержимого страницы, чтобы убе-

диться, что подсказки сохраняют своё положение и остаются видимыми и доступными. Тестируется, что подсказки адаптируются к изменению размера окна и не оказываются за пределами видимой области экрана, независимо от устройства или настроек пользователя.

Быстродействие. При проверке необходимо убедиться, что их появление и скрытие не замедляют работу приложения и не влияют на общую производительность, особенно когда речь идёт о динамически генерируемых подсказках¹², которые создаются в реальном времени в зависимости от контекста. Обращаем внимание, чтобы подсказки отображались без заметных задержек при взаимодействии с элементами интерфейса.

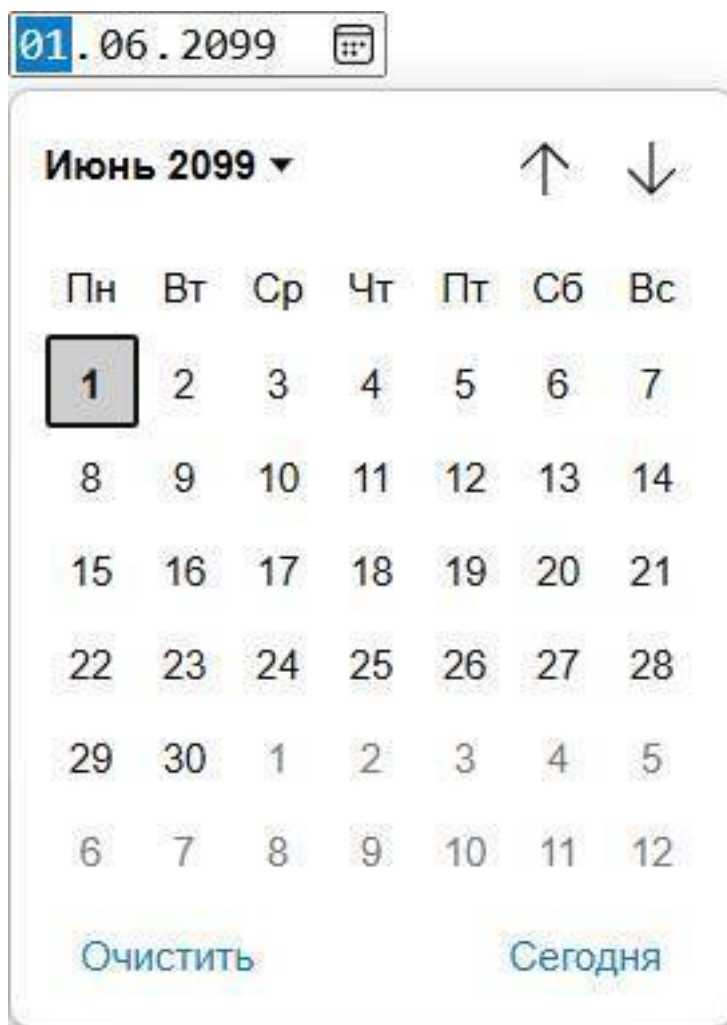
Разнообразные контексты использования. При анализе работы подсказок важно убедиться, что они корректно работают с различными типами элементов интерфейса, такими как кнопки, ссылки, формы ввода, и отображаются в соответствии с их предназначением. Тестирование охватывает разные сценарии использования: простые подсказки с краткими пояснениями, более сложные информационные подсказки с детальным содержанием, а также подсказки, содержащие динамически изменяющиеся данные. Важно убедиться, что подсказки адаптируются к любому контексту, не вызывают ошибок в работе интерфейса и остаются удобными и понятными для пользователя.

Обработка длинного текста. При тестировании обработки длинного текста в всплывающих подсказках необходимо убедиться, что такие тексты корректно отображаются и не нарушают визуальное оформление и позиционирование подсказок. Проверяется, что длинные тексты автоматически переносятся на новые строки без нарушения структуры и читаемости, не выходя за границы окна подсказки. Важно убедиться, что подсказка остаётся визуально аккуратной и не перекрывает другие элементы интерфейса, независимо от длины текста. Тестируется, как подсказка адаптируется к содержимому: размеры подсказки должны изменяться динамически, чтобы вместить весь текст, но при этом оставаться удобными для восприятия.

¹² Динамически генерируемая подсказка – это подсказка, которая появляется на экране в зависимости от того, что делает пользователь в данный момент. Например, мы начинаем вводить текст в поле поиска на сайте, и сайт сразу же показывает предложения или варианты поиска – это и есть динамически генерируемая подсказка. Она «подстраивается» под действия пользователя в реальном времени.

Тестирование календарей и элементов выбора даты

Тестирование календарей и элементов выбора даты важно для обеспечения удобства и точности ввода дат в приложениях. Эти элементы интерфейса используются во множестве сценариев, включая бронирование, планирование и установку сроков.



Приступим к рассмотрению ключевых аспектов, важных для тестирования.

Точность выбора даты. Необходимо убедиться, что пользователь может выбрать нужную дату, включая день, месяц и год, если эти параметры доступны. Проверяется, что элементы управления календарём, такие как стрелки для переключения месяцев, выпадающие списки или поля ввода, работают корректно и позволяют легко изменять даты. Тестируется, что все даты в календаре корректно отображаются и соответствуют реальным дням недели и что нет ошибок в расчёте числа дней в каждом месяце, особенно в случае високосных годов и месяцев с разным количеством дней. Важно убедиться, что интерфейс выбора дат удобен и понятен, с чётким визуальным выделением выбранной даты и возможностью изменить выбор при необходимости.

Ограничения диапазона дат. Проверяем, что все заданные ограничения работают корректно и предотвращают выбор недопустимых дат. Например, если для бронирования запрещено выбирать прошедшие даты, проверяется, что такие даты недоступны для выбора: они могут быть визуально затенены, неактивны для нажатия или отсутствовать. Также тестируются

ограничения на будущее, если применимо, чтобы пользователи не могли выбрать даты за пределами разрешённого интервала, например, бронирование только на ближайшие 6 месяцев. Важно убедиться, что календарь правильно отображает и применяет все настройки ограничений, информируя пользователя об ошибках выбора, если такие ограничения нарушены. Проверяются как интерфейсные блокировки, так и выбранные даты на стороне логики приложения.

Множественный выбор и диапазоны. При тестировании календарей и элементов выбора даты с поддержкой множественного выбора или выбора диапазона дат необходимо проверить корректность и удобство этих функций. Если пользователю можно выбрать несколько отдельных дат, проверяется, что даты можно легко и точно выделить и снять выделение без ошибок. Если календарь поддерживает выбор диапазона, необходимо убедиться, что выбор начала и конца диапазона интуитивно понятен, обычно через выделение двух крайних дат, и что все промежуточные даты правильно включаются в выбор. Визуальное отображение выбранных дат и диапазонов должно быть чётким и легко различимым, чтобы пользователь всегда понимал, какие даты активны. Тестируется, что при изменении выбора календарь правильно обновляет отображение.

Формат даты. Формат отображения выбранной даты должен соответствовать локализации пользователя или настройкам, заданным в приложении. Формат даты может отличаться в зависимости от региона: например, в определённых странах распространены следующие форматы отображения дат: «DD.MM.YYYY», «MM/DD/YYYY», «DD/MM/YYYY», «YYYY/MM/DD» и так далее. Проверяется, что календарь и элементы выбора даты отображают даты в корректном формате в соответствии с настройками локализации. Важно также убедиться, что при изменении языковых или региональных настроек приложения формат даты автоматически адаптируется к новым параметрам, сохраняя правильное представление. Тестируется, что при вводе даты вручную формат ввода соответствует отображаемому, а некорректные форматы распознаются и вызывают ошибки или предупреждения.

Некорректный ввод дат. При проверке обработки некорректного ввода дат, таких как 30 февраля или 31 апреля, необходимо убедиться, что приложение корректно распознаёт и обрабатывает подобные ошибки. Проверяется, что при вводе пользователем несуществующих дат приложение мгновенно идентифицирует их как некорректные и отображает соответствующее сообщение об ошибке, информирующее пользователя о неверном вводе. Важно, чтобы интерфейс не позволял сохранять или подтверждать такие даты, блокируя дальнейшие действия до исправления ошибки. Тестируется реакция приложения на различные типы некорректного ввода: неправильное количество дней в месяце, ошибки в формате, такие как использование недопустимых символов или букв вместо чисел.

Часовые пояса. Для приложений, работающих с пользователями из разных часовых поясов, даты и времена должны корректно учитывать часовой пояс пользователя. Проверяется, что приложение автоматически определяет часовой пояс на основе настроек устройства или пользователя и корректно отображает даты в соответствии с часовым поясом. Например, события, запланированные в одном часовом поясе, должны корректно отображаться в другом с учётом разницы во времени. При пересчёте времени между часовыми поясами важно учитывать переходы на летнее или зимнее время, чтобы избежать ошибок в расчёте времени. Также необходимо убедиться, что при ручном выборе или изменении часового пояса все связанные с ним даты автоматически обновляются в интерфейсе, отражая новые данные.

Интеграция с другими элементами формы. При тестировании интеграции элементов выбора даты с другими элементами формы необходимо убедиться, что взаимодействие между ними происходит корректно и не вызывает ошибок. Проверяется, что выбранная дата правильно передаётся в форму и сохраняется при отправке, без сбоев или потери данных. Особое внимание уделяется ситуациям, когда выбор даты влияет на другие поля формы, напри-

мер, когда выбранная дата автоматически обновляет диапазоны доступных дат в других полях (например, при выборе даты начала и окончания события). Важно убедиться, что зависимости работают правильно и связанные элементы обновляются без ошибок. Тестируется, что при изменении выбранной даты соответствующие поля формы корректно реагируют: блокируются или активируются другие даты, обновляются значения и подсказки и корректно срабатывают проверки на допустимые диапазоны.

Тестирование пагинации

Тестирование пагинации (постраничной навигации) крайне важно для приложений, предоставляющих пользователям большие объёмы данных, такие как каталоги товаров, списки статей или поисковые результаты. Пагинация помогает организовать содержимое, делая его более доступным и удобным для восприятия. Примеры:



Страницы ▾ : 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 След.

Разберём основные моменты, имеющие значение при тестировании пагинации.

Корректность переходов. Необходимо убедиться, что переходы между страницами работают корректно и соответствуют ожиданиям пользователя. Проверяется, что кнопки для перехода на следующую и предыдущую страницы правильно функционируют. Особое внимание уделяется проверке пагинации в условиях большого количества страниц, когда приложение должно корректно обновлять номера страниц и не допускать ошибок в отображении.

Соответствие номера страницы и содержимого. Проверяем, что каждая страница содержит правильный набор данных и что данные не повторяются или не пропускаются между страницами. Проверяется, что элементы, отображаемые на второй странице, не включают содержание с первой страницы и что каждый переход между страницами загружает именно тот набор данных, который должен отображаться на выбранной странице. Особое внимание уделяется тому, чтобы при навигации вперёд и назад данные оставались консистентными и не изменялись. Также проверяется, что при изменении настроек отображения, таких как количество элементов на странице, пагинация корректно пересчитывается и отображает соответствующий набор данных для каждой страницы, поддерживая непрерывность и точность информации.

Ограничения и ограниченный доступ. При тестировании пагинации в условиях ограниченного доступа к некоторым страницам необходимо убедиться, что приложение корректно обрабатывает ограничения и не позволяет пользователю переходить на недоступные страницы. Проверяется, что при попытке перейти на страницу с ограниченным доступом пользователь получает соответствующее уведомление или перенаправление и что сама кнопка или ссылка на такую страницу может быть заблокирована, скрыта или неактивна в зависимости от настроек приложения. Важно удостовериться, что пагинация правильно адаптируется к доступным страницам, корректно отображая номера страниц и их порядок, чтобы не было «пустых» переходов или ошибок. Тестируется правильная работа системы прав доступа, чтобы пользователи видели только те страницы, к которым у них есть разрешение, и не могли обойти ограничения через прямые ссылки или изменение URL.

Динамическая пагинация¹³. При анализе работы динамической пагинации, где содержимое страниц загружается по мере прокрутки или по запросу (например, при нажатии кнопки

¹³ Динамическая пагинация – это когда при прокрутке страницы вниз новые данные подгружаются автоматически, создавая впечатление, что страница бесконечная. Например, если на сайте десятки тысяч статей, то они будут подгружаться по

«Загрузить ещё»), необходимо убедиться, что новое содержимое подгружается. Проверяется, что данные загружаются плавно, без заметных пауз, подвисаний или ошибок, и что интерфейс приложения остаётся отзывчивым во время загрузки данных. Важно убедиться, что при динамическом добавлении новых элементов не происходит дублирования данных или пропуска элементов, а последовательность отображения сохраняется правильной. Проверяется, что динамическая пагинация корректно реагирует на пользовательские действия, такие как быстрая прокрутка или повторное нажатие на кнопку загрузки, и что данные подгружаются только один раз, без создания лишних запросов на сервер.

Видимость и доступность. Проверяем, что элементы пагинации заметны и удобны для взаимодействия на всех страницах, где используются. Элементы пагинации, такие как кнопки перехода или номера страниц, должны быть визуально выделены и находиться в привычных местах интерфейса, например, в нижней части списка или таблицы, чтобы пользователи могли без проблем их обнаружить. В случае динамической пагинации необходимо проверить наличие прогресс-бара или индикатора загрузки, который появляется при быстрой прокрутке страницы, если данные не успевают загружаться вовремя. Этот индикатор помогает пользователю понять, что происходит загрузка данных, и предотвращает возможные недоразумения, когда содержимое не появляется мгновенно.

Оптимизация загрузки. Для приложений с большим объёмом данных убеждаемся, что пагинация работает эффективно и не вызывает избыточных запросов к серверу или загрузки неактуальных данных. Проверяется, что запросы на сервер отправляются только при необходимости, то есть при переходе на новую страницу или при динамической загрузке содержимого, и что нет дублирования запросов при повторных переходах на одну и ту же страницу. Важно убедиться, что при навигации назад или вперёд приложение использует уже загруженные данные из кеша, а не запрашивает их повторно, снижая нагрузку на сервер и ускоряя отклик.

Тестирование граничных значений. При тестировании граничных значений в пагинации особое внимание уделяется корректности работы при переходах на первую и последнюю страницы, особенно в ситуациях, когда количество страниц может изменяться, например, при обновлении данных или изменении фильтров. Проверяется, что кнопки или ссылки для перехода на первую и последнюю страницы работают правильно и не вызывают ошибок. Важно убедиться, что при уменьшении общего количества страниц, например, после удаления данных или изменения условий поиска, система корректно обновляет пагинацию и не позволяет перейти на уже несуществующие страницы. Также тестируется поведение приложения при переходах с крайних страниц при изменении настроек количества отображаемых элементов, чтобы убедиться, что пагинация адаптируется и корректно пересчитывает доступные страницы. Особое внимание уделяется обработке ситуаций, когда пользователь находится на последней странице, а данные обновляются, чтобы система не оставляла пользователя на пустой странице или не вызывала ошибок отображения.

Тестирование загрузки файлов

Тестирование загрузки файлов через приложение требует тщательной проверки определённых аспектов, чтобы убедиться, что процесс работает корректно с разными сценариями использования. Под загрузкой файлов понимается отправка файлов на сервер через приложение.

Минимальный и максимальный размер файла. Необходимо убедиться, что приложение корректно обрабатывает файлы как минимально, так и максимально допустимого к загрузке размера, а также корректно реагирует на превышение установленных ограничений. Проверяется, что файлы минимального размера (например, пустые файлы или файлы с минимальным содержимым) успешно загружаются и распознаются приложением без ошибок. Для проверки максимального размера загружаются файлы, соответствующие верхнему пределу допустимого диапазона, чтобы убедиться, что система справляется с их обработкой и загрузкой без снижения производительности. Особое внимание уделяется файлам, которые превышают допустимый размер: такие файлы не должны загружаться, и приложение должно выводить пользователю сообщение об ошибке или предупреждение о превышении размера. Тестируется также, что интерфейс не зависает при загрузке файлов.

Поддерживаемые форматы файлов. При тестировании загрузки файлов различных форматов необходимо убедиться, что приложение корректно обрабатывает поддерживаемые форматы и блокирует загрузку неподдерживаемых. Проверяется, что файлы допустимых форматов, таких как изображения (JPG, PNG), документы (PDF, DOCX), таблицы (XLSX) и другие, загружаются без ошибок, отображаются правильно. Тестируется, что приложение корректно распознает каждый из поддерживаемых форматов и выполняет необходимые действия, такие как предварительный просмотр, редактирование или обработка файла. Особое внимание уделяется проверке неподдерживаемых форматов: при попытке загрузить файл неподдерживаемого типа (например, EXE, TIFF, или ZIP) приложение должно заблокировать загрузку и вывести пользователю сообщение. Сообщение должно чётко указывать причину отказа, например, «Формат файла не поддерживается», и по возможности предоставлять список поддерживаемых форматов. Это помогает пользователю быстро понять, в чем проблема, и избежать ошибок при загрузке файлов.

Корректность загруженных файлов. Проверяем, что файлы загружаются полностью и без искажений, сохраняя своё первоначальное содержимое. Проверяется, что при загрузке файлы передаются на сервер или сохраняются в системе без потерь данных, без изменения формата и содержания. Особое внимание уделяется проверке целостности больших файлов, например, видео, архивов или объёмных документов, чтобы убедиться, что они полностью загружены и не содержат ошибок, связанных с повреждением данных при передаче. После завершения загрузки файлы проверяются на соответствие исходным: проводится сравнение размеров, контрольных сумм (если это возможно) и визуальная проверка содержимого, чтобы убедиться, что загруженный файл идентичен исходному. Это гарантирует, что процесс загрузки надёжен и не приводит к повреждению данных.

Скорость загрузки. При тестировании скорости загрузки файлов важно проверить, как приложение справляется с загрузкой в условиях различного качества интернет-соединения, включая медленный интернет и нестабильное соединение. Тестируется поведение приложения при нормальной, медленной и нестабильной скорости интернета, чтобы убедиться, что загрузка файлов продолжается корректно, без зависаний или произвольных прерываний. Особое внимание уделяется длительным загрузкам больших файлов: приложение должно оставаться отзывчивым, показывать индикаторы прогресса и информировать пользователя о текущем статусе процесса, что особенно важно при медленной скорости. Также тестируется наличие

сообщений или индикаторов, предупреждающих о возможных задержках, чтобы пользователи понимали, что загрузка продолжается, и могли принять соответствующие меры.

Прерывание загрузки. При анализе поведения приложения при прерывании загрузки, например, из-за отключения интернета или временного сбоя соединения, важно проверить, как приложение реагирует на такие ситуации. Тестируем, что при обрыве загрузки пользователь получает соответствующее уведомление или сообщение об ошибке, информирующее о причинах остановки процесса. Важно убедиться, что система предлагает возможность возобновления загрузки с места прерывания, если это предусмотрено, чтобы не начинать процесс заново, особенно для больших файлов. Проверяется, что механизм восстановления работает корректно и стабильно, сохраняя ранее загруженные данные и продолжая процесс без потери информации. Если возобновление невозможно, приложение должно корректно завершить процесс с указанием причины сбоя, предлагая пользователю повторить попытку позже, при этом интерфейс не должен зависать.

Отображение процесса загрузки. Необходимо убедиться, что прогресс-бар или индикатор загрузки точно и наглядно отражают текущее состояние процесса. Прогресс-бар должен показывать реальное соотношение загруженного и оставшегося объёма данных, обновляясь плавно и без скачков, чтобы пользователь мог легко понять, на каком этапе находится загрузка. Если загрузка идёт в несколько этапов (например, подготовка, передача, обработка), индикатор должен чётко показывать переходы между этими этапами, чтобы пользователь знал, что именно происходит. Важно проверить, что при медленной загрузке индикатор не «зависает», а продолжает показывать прогресс, пусть и медленный, а при ошибках или прерываниях информирует об этом, не оставаясь на месте.

Валидация файлов на сервере. Проверяем, что приложение выполняет проверку загружаемых файлов на безопасность, предотвращая загрузку вредоносного кода или нежелательного содержимого. Проверяется, что сервер автоматически сканирует файлы на наличие вирусов, скрытых скриптов или других опасных элементов, используя встроенные средства безопасности или интегрированные антивирусные решения. Также важно удостовериться, что проверяются типы файлов и их содержимое, чтобы исключить маскировку вредоносного ПО под допустимые форматы, например, под видом изображений или документов. В случае обнаружения угрозы приложение должно мгновенно заблокировать загрузку и уведомлять пользователя о проблеме. Тестируется также, что все проверки выполняются на серверной стороне, чтобы избежать манипуляций с валидацией на уровне клиента, обеспечивая надёжную защиту от загрузки опасных файлов.

Ограничение доступа. При тестировании ограничения доступа к загруженным файлам необходимо убедиться, что приложение контролирует доступ к файлам через механизмы аутентификации¹⁴ и авторизации¹⁵ и файлы доступны только тем пользователям, которые имеют соответствующие права, и защищены от скачивания или просмотра посторонними. Тестируется корректная работа прав доступа: при попытке неавторизованного пользователя открыть или скачать файл система должна выдать сообщение об ошибке или перенаправить на страницу авторизации без предоставления доступа к содержимому. Особое внимание уделяется проверке прямых ссылок: файлы не должны быть доступны через прямые URL без проверки прав доступа, что предотвращает утечки данных.

¹⁴ Аутентификация – это процесс подтверждения личности пользователя. На данном этапе проверяется, что пользователь – действительно тот, за кого себя выдаёт. При аутентификации используются различные методы для проверки личности, например, пароль, биометрические данные или электронные ключи.

¹⁵ Авторизация – это процесс, который определяет, какие действия или ресурсы доступны аутентифицированному пользователю. Авторизация определяет, что пользователь может делать в системе после того, как его личность подтверждена. Например, после успешной аутентификации приложение может разрешить пользователю доступ к определённым файлам.

Конец ознакомительного фрагмента.

Текст предоставлен ООО «Литрес».

Прочитайте эту книгу целиком, [купив полную легальную версию](#) на Литрес.

Безопасно оплатить книгу можно банковской картой Visa, MasterCard, Maestro, со счета мобильного телефона, с платежного терминала, в салоне МТС или Связной, через PayPal, WebMoney, Яндекс.Деньги, QIWI Кошелек, бонусными картами или другим удобным Вам способом.