Лекция 7

Тестирование Web

Тестирование Web-приложений

- Функциональное тестирование.
- Тестирование безопасности.
- Нагрузочное тестирование.
- Кроссбраузерное тестирование.

1. Функциональное тестирование

рассматривает заранее указанное поведение и основывается на анализе спецификаций функциональности компонента или системы.

Функциональные тесты основываются на функциях, выполняемых системой, и могут проводиться на всех уровнях тестирования (компонентном, интеграционном, системном, приемочном).

Как правило, описываются в требованиях, функциональных спецификациях или в виде случаев использования системы.

2. Тестирование безопасности

- это стратегия тестирования, используемая для проверки безопасности системы, а также для анализа рисков, связанных с обеспечением целостного подхода к защите приложения, атак хакеров, вирусов, несанкционированного доступа к конфиденциальным данным.

Принципы безопасности программного обеспечения:

- Конфиденциальность - ограничение доступа к ресурсу некоторой категории пользователей или, другими словами, при каких условиях пользователь авторизован получить доступ к данному ресурсу.
- Целостность и доверие - ожидается, что ресурс будет изменен только соответствующим способом определенной группой пользователей.

- Повреждение и восстановление - в случае, когда данные повреждаются или неправильно меняются авторизованным или не авторизованным пользователем, Вы должны определить, насколько важной является процедура восстановления данных.
- Доступность - требования о том, что ресурсы должны быть доступны авторизованному пользователю, внутреннему объекту или устройству. Как правило, чем более критичен ресурс, тем выше уровень доступности должен быть.

Виды уязвимостей в безопасности ПО

- XSS (Cross-Site Scripting)
- XSRF / CSRF (Request Forgery)
- Code injections (SQL, PHP, ASP)
- Server-Side Includes (SSI) Injection
- Authorization Bypass

- **XSS** - это вид уязвимости программного обеспечения (Web приложений), при которой на генерированной сервером странице выполняются вредоносные скрипты с целью атаки клиента.
- **XSRF / CSRF** - это вид уязвимости, позволяющий использовать недостатки HTTP протокола (схема работы злоумышленников: ссылка на вредоносный сайт устанавливается на странице, пользующейся доверием у пользователя, при переходе по вредоносной ссылке выполняется скрипт, сохраняющий личные данные пользователя (пароли и т.д.), либо отправляющий СПАМ сообщения от лица пользователя, либо изменяет доступ к учетной записи пользователя для получения полного контроля над ней.

- **Code injections** - это вид уязвимости, при котором становится возможно осуществить запуск исполняемого кода с целью получения доступа к системным ресурсам, несанкционированного доступа к данным либо выведения системы из строя.
- **Server-Side Includes Injection** - это вид уязвимости, использующий вставку серверных команд в HTML код или запуск их напрямую с сервера.
- **Authorization Bypass** - это вид уязвимости, при котором возможно получить несанкционированный доступ к учетной записи или документам другого пользователя.

Примеры тестирования ПО на предмет уязвимости с системе безопасности

- **XSS**

Сами по себе XSS атаки могут быть очень разнообразными. Злоумышленники могут попытаться украсть ваши куки, перенаправить вас на сайт, где произойдет более серьезная атака, загрузить в память какой-либо вредоносный объект и т.д., всего навсего **разместив вредоносный скрипт** у вас на сайте.

Например скрипт, выводящий на экран ваши куки, или скрипт делающий редирект на зараженную страницу, или создающий вредоносный объект с вирусом и т.п.

- **XSRF / CSRF**

Наиболее частыми CSRF атаками являются атаки использующие HTML <IMG> тэг или Javascript объект image. Чаще всего, атакующий добавляет необходимый код в электронное письмо или выкладывает на веб-сайт таким образом, что, при загрузке страницы осуществляется запрос, выполняющий вредоносный код.

- **Code injections**

Вставки исполняемого кода

- **Server-Side Includes (SSI) Injection**

В зависимости от типа операционной системы команды могут быть разными. Например команда, которая выводит на экран список файлов в OS Linux

- **Authorization Bypass**

Пользователь А может получить доступ к документам пользователя Б. Допустим, есть реализация, где, при просмотре своего профиля, содержащего конфеденциальную информацию, в URL страницы передается userID, в данном случае, есть смысл попробовать подставить вместо своего userID номер другого пользователя. И если Вы увидите его данные, значит Вы нашли дефект.

Вывод

- Примеров уязвимостей и атак существует огромное количество. Даже проведя полный цикл тестирования безопасности, нельзя быть на 100% уверенным, что система по-настоящему обезопасена. Но можно быть уверенным в том, что процент несанкционированных проникновений, краж информации и потерь данных будет в разы меньше, чем у тех, кто не проводил тестирования безопасности.
- С развитием сетевых технологий и интернета взаимодействие разных систем, сервисов и приложений друг с другом приобрело значительную актуальность, так как любые связанные с этим проблемы могут привести к падению авторитета компании, что, как следствие ,повлечет за собой финансовые потери.

3. Нагрузочное тестирование (тестирование производительности)

- это автоматизированное тестирование, имитирующее работу определенного количества бизнес пользователей на каком-либо общем (разделяемом ими) ресурсе.

Основные виды нагрузочного тестирования

- Тестирование производительности (Performance testing)
- Стрессовое тестирование (Stress Testing)
- Объемное тестирование (Volume Testing)
- Тестирование стабильности или надежности (Stability / Reliability Testing)

производительности

Задачей тестирования производительности является определение масштабируемости приложения под нагрузкой, при этом происходит:

- измерение времени выполнения выбранных операций при определенных интенсивностях выполнения этих операций;
- определение количества пользователей одновременно работающих с приложением;
- определение границ приемлемой производительности при увеличении нагрузки (при увеличении интенсивности выполнения этих операций);
- исследование производительности на высоких, предельных, стрессовых нагрузках.

Стрессовое тестирование

Стрессовое тестирование позволяет проверить, насколько приложение и система в целом работоспособны в условиях стресса и также оценить способность системы к регенерации, т.е. к возвращению к нормальному состоянию после прекращения воздействия стресса.

Стрессом, в данном контексте, может быть повышение интенсивности выполнения операций до очень высоких значений или аварийное изменение конфигурации сервера. Также одной из задач при стрессовом тестировании может быть оценка деградации производительности, таким образом, цели стрессового тестирования могут пересекаться с целями тестирования производительности.

Объемное тестирование

Задачей объемного тестирования является получение оценки производительности при увеличении объемов данных в базе данных приложения, при этом происходит:

- измерение времени выполнения выбранных операций при определенных интенсивностях выполнения этих операций;
- может производиться определение количества пользователей одновременно работающих с приложением.

Тестирование стабильности или надежности

Задачей тестирования стабильности (надежности) является проверка работоспособности приложения при длительном (многочасовом) тестировании со средним уровнем нагрузки.

Время выполнения операций может играть в данном виде тестирования второстепенную роль. При этом, на первое место выходит отсутствие утечек памяти, перезапусков серверов под нагрузкой и другие аспекты влияющие именно на стабильность работы.

4. Кросс-браузерное тестирование

представляет собой процесс тестирования веб-приложений в нескольких браузерах.

В настоящее время пользователи имеют широкий выбор браузеров для доступа к веб-приложениям, поэтому в современных условиях стало крайне важным выполнять кросс-браузерное тестирование. В разных браузерах компоненты приложений могут вести себя по-разному.

Такое поведение приложения может быть вызвано рядом факторов:

- Разработанное веб-приложение может быть не адаптивно под тот или иной браузер или его версию.
- Неверно были применены стандарты, по которым разрабатывается сам браузер.
- Были допущены ошибки при разработке браузера.
- У пользователя установлен какой-либо плагин или надстройка, вызывающие ошибки веб-приложения.

Из-за разной работы браузеров или ошибок, допущенных в нем, могут возникать дефекты в Вашем продукте.

Часто встречающиеся дефекты:

- Верстка.
- Навигация.
- Ошибки JavaScript.

- **Верстка.**

Наиболее распространенная ошибка в различных браузерах. Разработчики часто создают приложение и проверяют его в одном, наиболее удобном дня них, браузере. Но у пользователей может быть установлена другая версия браузера, в котором «красивая картинка» разработчика может выглядеть совсем некрасиво у пользователя. Такие ошибки чаще всего не являются критичными, но неприятное впечатление о вашем продукте могут оставить.

- **Навигация.**

Бывают ситуации, когда в одном из браузеров ссылка не работает, как было запланировано, либо не работает вовсе. Такие ошибки могут негативно отразиться на Вашем продукте. Когда клиент не находит нужный раздел или не может перейти на другую страницу, чтобы завершить действие, это доставляет неудобства и раздражает. Как результат – потеря клиента.

- **Ошибки JavaScript.**

Такие ошибки имеют высокий приоритет. Неработоспособность JavaScript в одном из браузеров может привести к потере заказа, клиента, или к потере документа, например, если у вас система электронного документооборота; к невозможности создания заявки, если у вас система заявок и т.д.

Десктопные браузеры:

- Chrome
- Firefox
- IE
- Safari
- Edge
- Opera

Мобильные браузеры:

- Chrome
- Safari
- UC Browser
- Opera
- Samsung Internet
- Android

Движки браузеров:

- Trident - проприетарный движок Microsoft Internet Explorer.
- Gecko - открытый движок проекте MozillaFirefox.
- KHTML - разработан в рамках проекта KDE, послужил основой для WebKit.
- WebKit - движок для браузера Apple Safari и Google Chrome.
- Presto - проприетарный движок для браузера Opera до перехода на Blink.
- Blink - движок браузера Google Chrome с 28 версии и Opera с 15 версии. Является ответвлением WebKit.
- Edge - новый движок от компании Microsoft для браузера Microsoft Edge. Является ответвлением Trident.

Тестирование UI и верстки

UI (user interface — пользовательский интерфейс) — является точкой взаимодействия человека и продукта.

Дизайн кнопок, полей ввода и т.д. — это место, где пользователь взаимодействует с системой.

Дизайн пользовательского интерфейса (UI) — это дизайн точек взаимодействия, через которые пользователь может взаимодействовать с системой, подобно тому, как взаимодействие с рулем, педалями и приборной панелью используется для управления автомобилем .

Тестирование интерфейса пользователя осуществляется вместе со следующими видами тестирования (UI):

- Тестирование на соответствие стандартам графических интерфейсов.
- Тестирование с различными разрешениями экрана.
- Тестирование кроссбраузерности или совместимости с разными интернет браузерами и их версиями.
- Тестирование локализованных версий: точность перевода (мультиязычность, мультивалютность), проверка длины названий элементов интерфейса и т.д.
- Тестирование графического интерфейса пользователя на целевых устройствах (смартфоны, кпп, планшеты).

Основные элементы графического интерфейса:

- Окно (окно браузера, диалоговое окно, модальное окно, плавающее окно).
- Меню (главное, всплывающее, контекстное, системное).
- Виджеты/элементы управления/контролы (аккордеон, кнопка, радио-кнопка, чек-бокс, значок (иконка), список, панель инструментов, дерево, полоса прокрутки, ползунок, строка состояния, тултип (подсказка) и др.).
- Вкладка.
- Элементы взаимодействия: курсор мыши, текстовый курсор, поинтер (“ладошка”), курсор перетаскивания и др.

Основные проверки при тестировании UI:

- Расположение, размер, цвет, ширина, длина элементов; возможность ввода букв или цифр.
- Реализуется ли функционал приложения с помощью графических элементов.
- Размещение всех сообщений об ошибках, уведомлений (а также шрифт, цвет, размер, расположение и орфография текста).
- Читабелен ли использованный шрифт.
- Переходит ли курсор из текстового в поинтер при наведении на активные элементы, выделяются ли выбранные элементы.
- Выравнивание текста и форм.

- Качество изображений.
- Проверить расположение и отображение всех элементов при различных разрешениях экрана, а также при изменении размера окна браузера (проверить, появляется ли скролл).
- Проверить текст на орфографические, пунктуационные ошибки.
- Появляются ли тултипы (если есть необходимость).
- Унификация дизайна (цвета, шрифты, текст сообщений, названия кнопок и т.д.).

Тестирование Pixel Perfect

- проверка точного (пиксель в пиксель) соответствия сверстанного HTML-шаблона оригиналу (PSD-макету). Другими словами, если наложить “картинку” сверстанного HTML-шаблона на картинку оригинального PSD-макета, то обе картинки должны совпадать. Совместиться должны все элементы картинок: текст, изображения, графические элементы.

При проектировке качественного UI уделяется внимание не только внешнему виду интерфейса, но и его логической структуре, чтобы пользователь мог без лишних усилий, быстро и легко взаимодействовать с ним и добиваться необходимого результата.

Но, чтобы четко понимать, как создать качественный пользовательский интерфейс для конкретного продукта, необходимо изучать поведение, эмоции и реакцию пользователей при взаимодействии с данным продуктом, проводить тестирование, собирать данные.

Человек, взаимодействуя с какой-либо системой, испытывает ощущения и реагирует определенным образом в процессе ее использования. Это называется опытом взаимодействия, или UX.

User Experience (пользовательский опыт)

— ощущение, испытываемое пользователем во время использования цифрового продукта, в то время как User interface — это инструмент, позволяющий осуществлять интеракцию «пользователь — веб-ресурс». UX — это то, что чувствует и запоминает пользователь в результате использования программы, приложения или сайта. UX учитывается при разработке UI, создании информационной архитектуры, юзабилити тестировании. Определив целевую аудиторию и характеристики основного пользователя можно составить список требований к проекту.

Пример: предположим, мы едим сэндвич с сыром.

Ощущение, получаемые, когда едим бутерброд, можно считать UX, а ингредиенты сэндвича (хлеб, сыр, майонез и т. д.) ассоциируются с UI.

Сэндвич, сделанный из белого хлеба и сыра и майонеза с высоким содержанием жиров, на вкус почти также хорош, как бутерброд, состоящий из цельнозернового хлеба, низкокалорийного майонеза и нежирного сыра. Однако люди, стремящиеся к здоровому образу жизни, отвергнут первый сэндвич в пользу второго.

Итак, у нас есть хороший интерфейс в обоих случаях, но мы не провели пользовательское исследование (а это неотъемлемая часть UX), мы не знаем соотношения пользователей, которые будут/не будут использовать наш продукт, в результате чего мы теряем весомую часть целевой аудитории.

Процесс проектирования UX включает в себя исследование поведенческих паттернов и психологических реакций пользователей, разработку информационной архитектуры, дизайн взаимодействия (interaction design), дизайн пользовательского интерфейса, интерактивное прототипирование макета (interactive prototyping) и тестирование юзабилити (usability testing).

Дизайнеры пользовательского интерфейса должны обладать навыками в области визуального дизайна, иконографики и типографики, однако в перечень их служебных обязанностей не обязательно входит проведение пользовательских исследований или построение информационной архитектуры веб-ресурса.

А вот дизайнеры пользовательского опыта должны дополнительно еще и разбираться в исследованиях целевого рынка, information architecture и дизайне взаимодействий (что автоматически подразумевает базовое знание поведенческой психологии) и т. д.

Юзабилити (usability)

–дословно с английского означает: возможность использования или полезность. Юзабилити – это больше мера дружелюбности сайта или интерфейса программы, поскольку оно помогает сделать сайт понятным и естественным для пользователя.

Тестирование удобства пользования (Usability Testing) – это метод тестирования, направленный на установление степени удобства использования, обучаемости, понятности и привлекательности для пользователей разрабатываемого продукта в контексте заданных условий.

Этапы тестирования удобства использования пользовательского интерфейса:

- **1.Исследовательское** - проводится после формулирования требований и спецификаций к системе, а также после разработки прототипа интерфейса. Основная цель на этом этапе - выяснить, позволяет ли он с достаточной степенью эффективности решать задачи пользователя.

- **2. Оценочное** - проводится после разработки низкоуровневых требований и детализированного прототипа пользовательского интерфейса. Оценочное тестирование углубляет исследовательское и имеет ту же цель. На данном этапе уже проводятся количественные измерения характеристик пользовательского интерфейса: измеряются количество обращений к системе помощи по отношению к количеству совершенных операций, количество ошибочных операций, время устранения последствий ошибочных операций и т.п.

- **3. Валидационное** - проводится ближе к этапу завершения разработки. На этом этапе проводится анализ соответствия интерфейса программной системы стандартам, регламентирующим вопросы удобства интерфейса, проводится общее тестирование всех компонентов пользовательского интерфейса (с точки зрения конечного пользователя). Под компонентами интерфейса здесь понимается как его программная реализация, так и система помощи и руководство пользователя. Также на данном этапе проверяется отсутствие дефектов удобства использования интерфейса, выявленных на предыдущих этапах.

- **4. Сравнительное-** данный вид тестирования может проводиться на любом этапе разработки интерфейса. В ходе сравнительного тестирования сравниваются два или более вариантов реализации пользовательского интерфейса.

Из этого следует, что UI-тестирование, предполагает под собой тестирование на основании требований к внешнему виду пользовательского интерфейса и формам взаимодействия с пользователем. На какие требования стоит обращать внимание при UI-тестировании:

- Требования к размещению элементов управления на экранных формах.
- Требования к содержанию и оформлению выводимых сообщений.
- Требования к форматам ввода.
- Требования к реакции системы на ввод пользователя.
- Требования к времени отклика на команды пользователя.

Важно обращать внимание на:

- Простоту использования сайта или интерфейса.
- Эффективность использования.
- Запоминаемость.
- Ошибки, их количество и серьезность.
- Удовлетворение пользователя (субъективное).

GUI тестирование включает:

- Тестирование пользовательского интерфейса (UI)
- Тестирование удобства использования (Usability Testing)
- Compatibility testing (тестирование совместимости)

Тестирование пользовательского интерфейса (UI):

– тестирование, выполняемое путем взаимодействия с системой через графический интерфейс пользователя:
- навигация;
- цвета, графика, оформление;
- содержание выводимой информации;
- поведение курсора и горячие клавиши;
- отображение различного количества данных (нет данных, минимальное и максимальное количество);
- изменение размеров окна или разрешения экрана.

Тестирование удобства использования (Usability Testing)

– тестирование с целью определения степени понятности, легкости в изучении и использовании, привлекательности программного продукта для пользователя при условии использования в заданных условиях эксплуатации:

- визуальное оформление;
- навигация;
- логичность.

Compatibility testing (тестирование совместимости)

– процесс тестирования для определения возможности взаимодействия программного продукта, проверка работоспособности приложения в различных средах (браузеры и их версии, операционные системы, их типы, версии и разрядность)

Виды тестов:

- Кросс-браузерное тестирование (различные браузеры или версии браузеров).
- Кросс-платформенное тестирование (различные операционные системы или версии операционных систем).

Прежде чем тестировать сервис, нужно разработать план тестирования юзабилити. Обычно он содержит следующие разделы:

- Цели и задачи теста.
- Вопросы исследования.
- Характеристики приложения.
- Метод тестирования.
- Список заданий.
- Испытательная среда, оборудование и логистика.
- Собираемые данные и меры оценки.
- Презентация отчета.

Методы оценки юзабилити разные у разных команд и тестировщиков.

Например, Дэвид Трэвис из User Focus предлагает такой план тестирования:

- Цели теста.
- Задачи, которые будет выполнять приложение.
- Тестовые документы (форма содержания, сценарий ориентации, анкеты до и после тестирования).
- Участники теста.
- Метод испытания.

Программное обеспечение, применяемое при тестировании UI

Для тестировщика могут быть полезны такие **расширения Chrome:**

- **Screen Ruler** - помогает измерять высоту, ширину и поля у объектов, просто накладывая поверх них линейку и перетаскивая ее в нужном направлении.
- **What Font** - позволяет проверять косметические дефекты вроде типа и размера шрифта на страничке. Это расширение позволяет легко искать шрифты. Просто наведите курсор на любой шрифт и оно покажет Вам, какой именно шрифт тут применен.

- **Color Zilla** - позволяет определить прямо из браузера, какой конкретно цвет используется на Вашей страничке. Полезное расширение, когда Вам нужно сверить реально используемые цвета со спецификацией.
- **Perfect Pixel** - замена Photoshop, позволяет прямо в браузере наложить полупрозрачное изображение поверх вашей странички и провести попиксельный анализ.
- **IE Tab** - один из самых популярных IE-эмуляторов. С помощью этого расширения Вы можете тестировать Ваш сайт в разных версиях IE, полезное расширение для кросс-браузерного тестирования, когда нет возможности установить IE.

- **Spell Checker** - расширение для проверки орфографии. Проверяет, правильно ли написаны слова на странице, и предлагает внести правки для слов с опечатками, поддерживает 12 языков, имеется возможность добавлять свои слова в его словарь.
- **Grammarly** – приложение, которое помогает с вычиткой текста, проверяет орфографию и грамматику и подсвечивает ошибки прямо в браузере.

Существуют профессиональные инструменты для тестирования пользовательского интерфейса, которыми пользуются чаще всего frontend-разработчики, а также они распространены для автоматизации данного вида тестирования:

- **FitNesse**:

Это инструмент для совместной разработки программного обеспечения. Он позволяет клиентам, тестировщикам и программистам изучить то, что должно сделать их программное обеспечение, и автоматически сравнить с тем, что программное обеспечение фактически делает.

- **iMacros**:

Это программа, которая позволяет сохранять и воспроизводить действия пользователя. Распространяется как в виде отдельного ПО, так и в виде расширений для браузеров Mozilla Firefox, Google Chrome и Internet Explorer. Расширение способно работать с сайтами, реализованными при помощи технологий Java, Flash, Flex, Ajax и Silverlight.