**Лекція 20. Особливості тестування веб-додатків , ігрових додатків**

ОСОБЛИВОСТІ ТЕСТУВАННЯ WEB-ДОДАТКІВ

**Визначення та основні складові веб-додатку**.

Веб-додатки – сфера, яка динамічно розвивається. Не усі підходи та методи, які застосовуються для тестування класичних додатків можуть бути використанні при тестуванні веб-додатків. На рисунку 1 наведена структура веб-додатку.

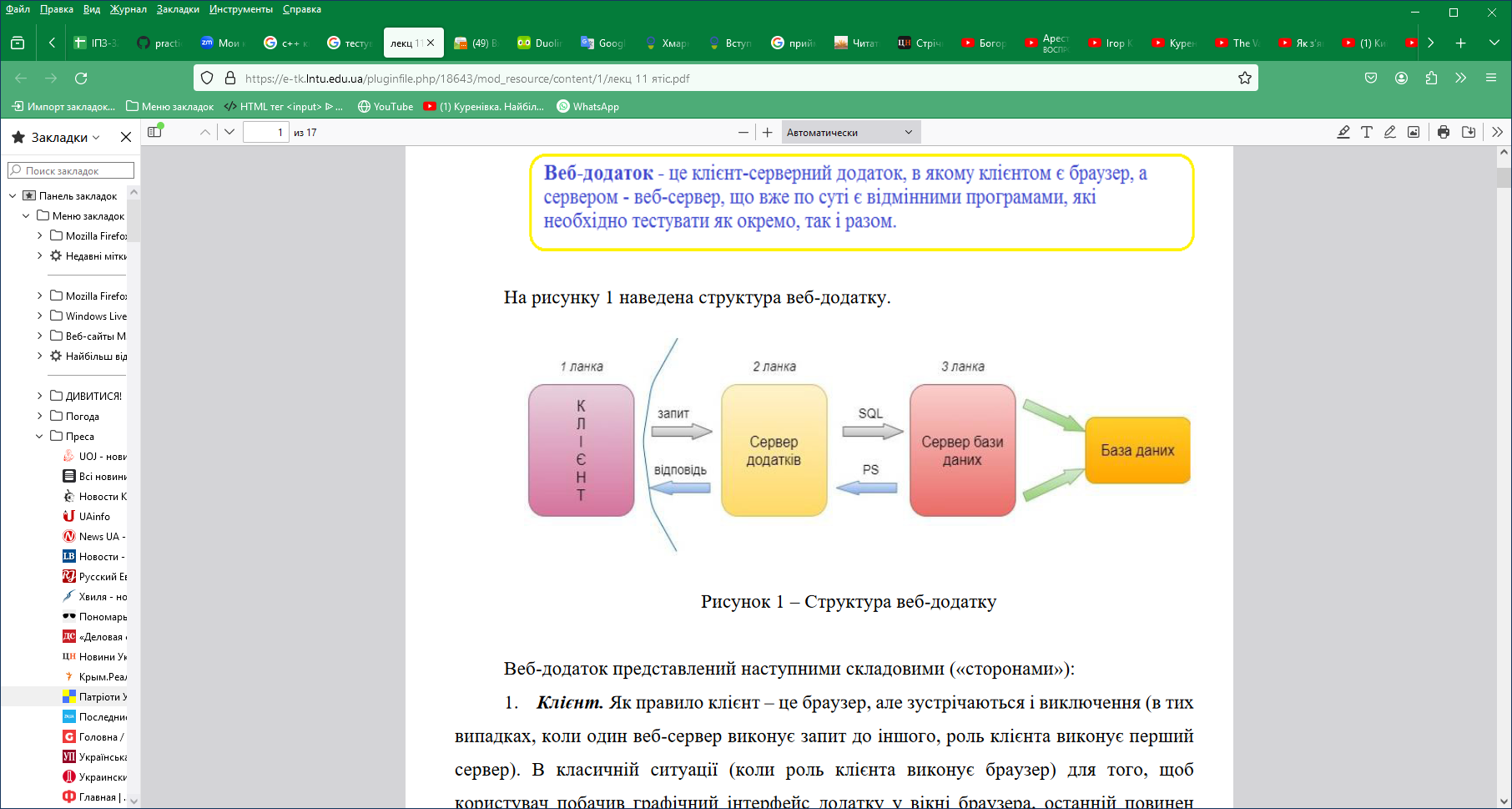


Рисунок 1 – Структура веб-додатку

Веб-додаток представлений наступними складовими («сторонами»):

1. ***Клієнт***. Як правило клієнт – це браузер, але зустрічаються і виключення (в тих випадках, коли один веб-сервер виконує запит до іншого, роль клієнта виконує перший сервер). В класичній ситуації (коли роль клієнта виконує браузер) для того, щоб користувач побачив графічний інтерфейс додатку у вікні браузера, останній повинен обробити отриману відповідь веб-сервера, в якій буде міститись інформація, реалізована із застосуванням HTML, CSS, JS (найбільш застосовувані технології). Саме ці технології «дають зрозуміти» браузеру, як саме необхідно «відтворити» все, що він отримав у відповіді.

2. ***Сервер***. Веб-сервер – це сервер, який приймає НТТР-запити від клієнтів та віддає їм НТТР-відповіді. Щоб уникнути можливої плутанини, відмітимо, що веб-сервером називають як програмне забезпечення, що виконує функції веб- сервера, так і безпосередньо комп’ютер, на якому це програмне забезпечення працює. Найбільш розповсюдженими видами програмного забезпечення веб- серверів є Apache, IIS та NGINX. На веб-сервері функціонує тестований додаток, який може бути реалізований із застосуванням самих різноманітних мов програмування: PHP, Python, Ruby, Java, Perl та інші.

3. ***База даних***. В класичній теорії мова іде про дві «сторони» веб-додатку, однак, якщо уважно подивитись на весь процес роботи додатку, ми можемо відповісти, що в алгоритмі роботи веб невидимо, але доволі активно приймає участь ще одна «сторона» -база даних. Фактично вона не є частиною веб- сервера, але більшість додатків просто не можуть виконувати всі покладені на них функції без неї, так як саме в базі даних зберігається вся динамічна інформація додатку (звітні, користувацькі дані та інші).

База даних – доволі широке поняття, котре використовується не лише в сфері інформаційних технологій. В даному контексті – це інформаційна модель, котра дозволяє впорядковано зберігати дані про об’єкт чи групу об’єктів, які володіють набором властивостей, котрі можна категоризувати. База даних функціонує під керівництвом систем керування базами даних (СУБД). Найбільш популярними СУБД є MySQL, MS SQL Server, PostgreSQL, Oracle (все веб-серверні).

Також існують вбудовані та файл-серверні СУБД. Для загального розвитку відмічу лише одну найбільш популярні вбудовану СУБД – SQLite, котра використовуються в деяких браузерах, Android API, Skype та інших відомих додатках. Взаємодія з перерахованими СУБД основана на спеціальній мові структурованих запитів – SQL.

Майже всі сучасні програми орієнтовані на роботу з мережею. Збереження даних веб-додатків здійснюється зазвичай на сервері, обмін інформацією відбувається по мережі. Коли ми бачимо помилку в мережевому середовищі, то досить часто важко впевнено сказати, де саме вона відбувалась, і тому режим роботи чи повідомлення про помилку, яке ми отримуємо, може бути результатом помилок, які виникли в різних частинах мережі.

Маючи багато спільного із тестуванням класичного додатку, тестування веб-орієнтованих додатків має свої особливості, пов’язані в першу чергу із середовищем функціонування. Маючи компонентні, структурні та технологічні особливості, веб-додаткам притаманні особливості режимів роботи, інсталяції, запуску, зупинки та видалення, а також формування інтерфейсів. Працюючи завжди з мережею та великою кількістю користувачів, веб-додатки мають на увазі під собою різноманітні права доступу для різних користувачів.

Логіка веб-додатку розподілена між сервером та клієнтом, зберігання даних здійснюється на сервері, обмін інформацією відбувається по мережі. Однією із переваг підходу є той факт, що клієнти не залежать від конкретної операційної системи користувача, тому веб-додатки є міжплатформеними сервісами. Те, що веб-розробки будуть продовжувати збільшувати темпи свого розвитку, підтверджується і набираючим силу «мейнстрімом»: все «переміщується» в хмару. Хмарні технології стають реальністю сучасного Інтернету: навіть колись звичні нам десктопні Word та Excel сьогодні представлені у вигляді веб-альтернатив від Microsoft. Виходячи із сказаного, можна стверджувати, що необхідність в деяких інженерах по забезпеченню якості, які спеціалізуються на веб- продуктах, буде лише збільшуватись.

2. Особливості тестування веб-додатків

1. Технологічні відмінності. Класичний додаток працює із використанням однієї чи сімейства споріднених технологій.

2. Структурні відмінності. Класичний додаток «монолітний». Складається з одного чи невеликої кількості модулів. Не використовує сервери бази даних, веб-сервери та інше. Веб-додаток – «багатокомпонентний». Складається з великої кількості модулів. Обов’язково використовує сервери бази даних, веб-сервери та сервери додатків.

3. Відмінності режимів роботи. Класичний додаток працює в режимі реального часу, тобто відомо про дії користувача одразу ж, як тільки вони були виконані. Веб-додаток працює в режимі «запит-відповідь», тобто відомо про деякий набір дій тільки після запиту на сервер (див.рис.2).

4. Відмінності формування інтерфейсу. Класичний додаток використовує для формування інтерфейсу користувача відносно усталені та стандартизовані технології. Веб-додатки використовують для формування користувацького інтерфейсу технології, які стрімко розвиваються, більшість яких конкурують між собою.

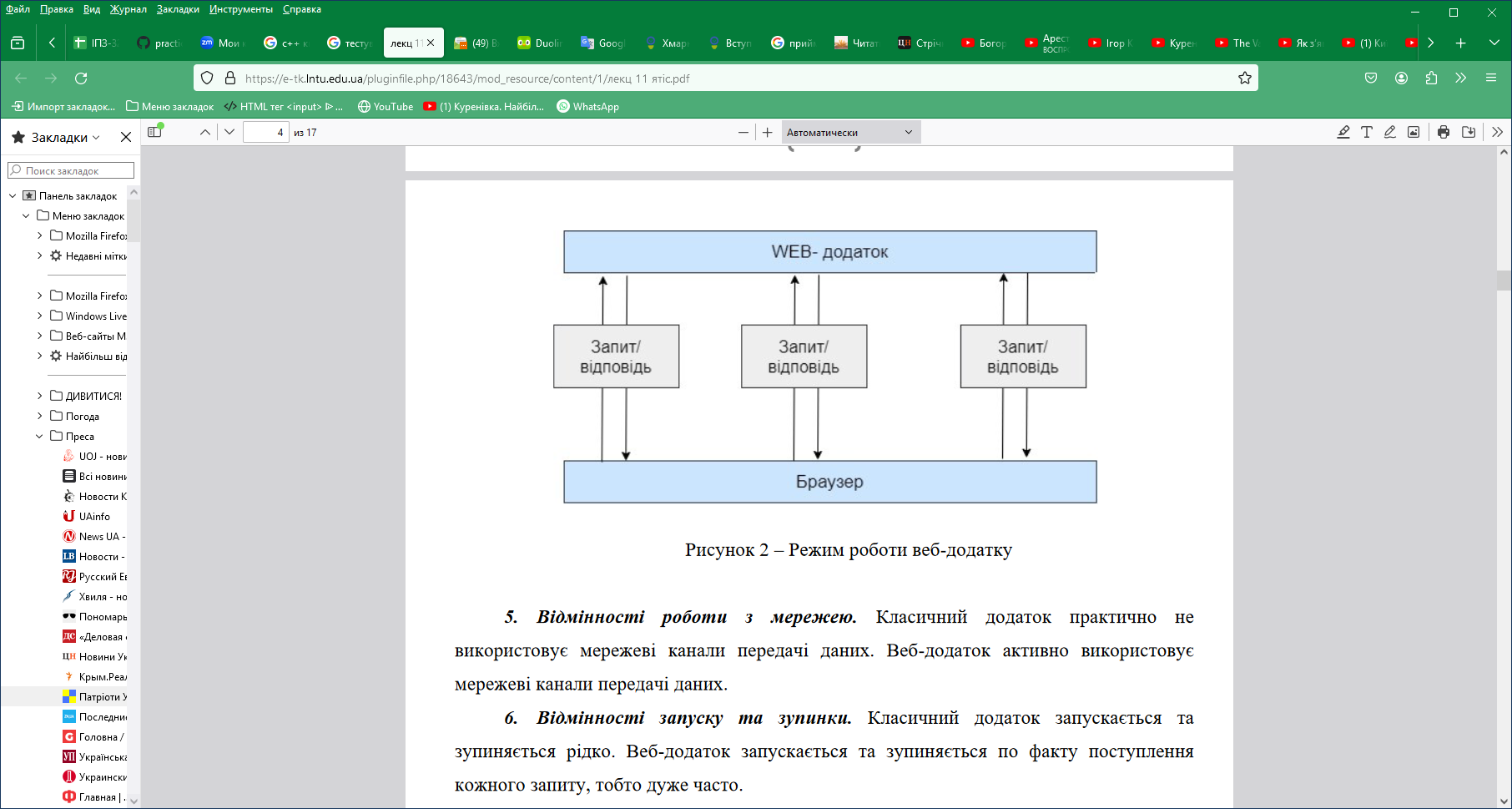


Рисунок 2 – Режим роботи веб-додатку

5. Відмінності роботи з мережею. Класичний додаток практично не використовує мережеві канали передачі даних. Веб-додаток активно використовує мережеві канали передачі даних.

6. Відмінності запуску та зупинки. Класичний додаток запускається та зупиняється рідко. Веб-додаток запускається та зупиняється по факту поступлення кожного запиту, тобто дуже часто.

7. Різниця в кількості користувачів. Класичний додаток: кількість користувачів, які одночасно використовують додаток, контрольована, обмежена та легко прогнозована. Веб-додаток: кількість користувачів, які одночасно використовують додаток, важкопрогнозована і може стрибкоподібно змінюватись в широких діапазонах.

8. Особливості збоїв та відмов. Класичний додаток: вихід із ладу тих чи інших компонент одразу стає очевидним. Веб-додаток: вихід з ладу деяких компонент дає непередбачуваний вплив на працездатність додатку в цілому.

9. Відмінності в інсталяції. Класичний додаток – процес інсталяції стандартизований та максимально орієнтований на широку аудиторію користувачів. Не потребує специфічних знань. Додавання компонент додатку виконується стандартним способом із використанням одного і того ж інсталятора. Веб-додаток – процес інсталяції часто недоступний кінцевому користувачеві. Інсталяція потребує специфічних знань. Процес зміни компоненти додатку не передбачає або потребує кваліфікації користувачів, інсталятор відсутній.

10. Відмінності в деінсталяції. Класичний додаток: процес деінсталяції стандартизований та виконується автоматично чи напівавтоматично. Веб-додаток: процес деінсталяції потребує специфічних знань для втручання адміністратора і часто поєднаний із зміною коду середовища функціонування додатку, бази даних, налаштування системної операційної системи.

11. Особливості середовища функціонування. Класичний додаток: середовище функціонування стандартизоване та не сильно впливає на функціонування додатку. Веб-додаток: середовище функціонування дуже різноманітне і може проявляти серйозний вплив на дієздатність як серверної, так і клієнтської частини.

3. Особливості архітектури: «під прицілом» клієнт. Ми зазначали клієнта як першу складову архітектури. В класичній ситуації клієнт представлений браузером, а тому питання тестування кросбраузерності досить актуальний. Ми також розглянемо тестування форм, які заповнюються та тексту, як основного джерела інформації, що отримується від клієнта.

*Кросбраузерність: різноманіття клієнтів*

Що ми розуміємо під тестуванням на кросбраузерність? Це перевірка на правильність (відповідність вимогам і стандартам) відображення та функціонування веб-додатку в різних браузерах та на різних операційних системах. В сучасному світі стандартизація приймає глобальні масштаби, а тому більшість популярних браузерів однаково обробляють код. При цьому необхідність в кросбраузерному тестуванні не зникає, так як далеко не всі проблеми вирішуються стандартизацією.

Перед початком робіт рекомендовано взнавати у відповідальних людей необхідний для тестованого програмного забезпечення набір браузерів. Можна самим зібрати статистику по цільовій аудиторії вашого продукту та обмежити кросбраузерне тестування набором найбільш популярних в цій аудиторії браузерів. В умовах регресійного тестування, коли обмежені часові та людські ресурси, також можна застосовувати практику розподілу браузерів: закріпляючи за кожним тестувальником певний браузер, ви зможете «покрити» більший відсоток кросбраузерних дефектів. Цей метод має і свої недоліки: він найбільш результатиний у великій команді тестувальників, але стає практично незастосовуваним при наявності одного тестувальника.

Що необхідно перевіряти при кросбраузерному тестуванні:

- функціональні можливості продукту, які реалізуються на стороні клієнта;

- правильність відображення елементів графіки;

- шрифти та розміри текстових символів;

- доступність та функціональність різноманітних форм, включаючи їх інтерактивність.

Тестування потребують всі основні (серед користувацької аудиторії) браузери, але особливу увагу необхідно приділити Internet Explorer, якщо він входить в їх кількість.

Саме в ньому дуже часто виникають проблеми, котрі відсутні в інших браузерах: так як в Internet Explorer додатково рекомендується звертати особливу увагу на масштабованість, фокус полів та роботу JS.

Окремо варто не забувати про різного роду валідаторів верстки. Навіть якщо у вас достатньо знань, щоб оцінити відповідність верстки стандартам, можна використати для цього автоматичні засоби і, проаналізувавши результат, вказати розробникам на самі серйозні «огріхи». Не варто забувати, що інколи валідатори звертають увагу на самі «дрібні дрібниці», котрі ніхто і ніколи виправляти не буде. Якщо ви і заносите баг-репорти на подібні зауваження, то зручніше буде зібрати їх в окремий документ та прикріпити до репорта. До такого роду «дрібниць» можна віднести всі можливі рекомендації, котрі не мають свого впливу на відображення та функціонування контенту.

*Веб-форми на стороні клієнта*

Однією із важливих складових інтернет-додатків є форми заповнення, взаємодію із якими користувач здійснює за допомогою все того ж клієнта. Однак, дані форми дуже часто слугують джерелом дефектів, котрі, розмістившись у «продакшині», можуть принести великі фінансові і репутаційні збитки компанії.

Як ж не пропустити дефекти у формах на продакшен? Розглянемо декілька простих кроків:

1. Ретельно перевіряємо обов’язковість заповнення полів та наявність відповідного маркування у них.

2. Заповнивши та надіславши форму, впевнюємось у тому, що з даними відбувається саме те, що заплановано. Якщо дані повинні бути внесені в базу даних, перевіряємо коректність внесення.

3. Використовуємо чіт-листи для тестування форм.

Чіт-лист (Cheat-sheet) – список перевірок, що повторюються. Чіт-листи складаються із ціллю їх наступного багаторазового використання. У зв’язку з цим такі списки складаються по відношенню до розповсюджених та часто зустрічних складових програмного забезпечення, з яким належить працювати неодноразово не лише в поточному проекті, але і в наступних.

Кожний окремий спеціаліст розширює перелік перевірок у своїх чіт- листах із досвідом. На сьогодні багато автоматизованих систем управління тест-кейсами також містять і функціонал підтримки списків чіт-листів, що в значній мірі спрощує їх створення, наступні підтримки та практичне застосування.

4. Перевіряємо чи виводяться зрозумілі користувачу інформаційні повідомлення про необхідність заповнення пустих полів після спроби надіслати форму.

5. Звертаємо увагу на реалізацію екранування символів в полях форм, які є потенційним джерелом вразливості для додатку та користувачів. Екранування повинно здійснюватись на рівні не лише клієнта, але і сервера, вимкнути який в клієнті досить просто (наприклад за допомогою спеціальних плагінів, котрі знімають всі можливі обмеження в декілька кліків).

6. Впевнюємось, що після заповнення форми, користувачу приходить підтверджувальний лист з вказуванням відповідного відправника, а саме тіло листа відповідає вимогам.

7. Використовуємо допоміжні спеціальні інструменти для тестування форм, наприклад Web Developer Toolbar.

Текст, як основне джерело інформації при роботі через клієнт Основною цінністю мережі Інтернет є те, що вона являється практично безмежним джерелом інформації. Частина цієї інформації представлена у вигляді текстів, з якими, знову ж таки, користувач взаємодіє за допомогою клієнту. Більшість веб-ресурсів в тому чи іншому об’ємі потребують перевірки текстів на предмет відсутності граматичних та друкарських помилок. Звісно, значимість цього тестування не така велика порівняно з функціональним напрямком, але нехтувати ним не потрібно. На практиці бувають серйозні друкарські помилки: на продакшині одного з великих інтернет-гіпермаркетів була допущена помилка, яка змінила назву міста на лайливе слово, яке було видно не лише мешканцям цього міста. Так, часу на вичитування всього тексту буває замало, тоді в таких випадках на допомогу приходять «SpellCheker» (програми для перевірки орфографії, онлайн або у вигляді плагінів для браузерів).

***Особливості архітектури: «під прицілом» сервер***

Тестування частини веб-додатку, розміщеної на веб-сервері, можна провести оминувши графічний(клієнтський) інтерфейс, однак це потребує від спеціаліста визначеного рівня знань та навиків технічного характеру, а також застосування додаткових інструментів. Розглянемо веб-сервер з точки зору тестування навантаження та інсталяції.

*Інсталяція на веб-сервер*

Отже, перед тим як почати тестувати, ми повинні встановити веб-додаток на веб-сервер. Відповідно тут є схожість з тестуванням десктоп додатків, але існує і відмінність в нюансах, котрі необхідно врахувати та протестувати, особливо якщо це стосується програмного забезпечення, розповсюджуваного для локальної інсталяції на веб-сервери користувачів.

В чому нюанси:

1. Більша частина веб-додатків потребує для інсталяції специфічних знань в адмініструванні операційних систем. Спробуйте встановити додаток на декількох веб-серверах. В оптимальному випадку це будуть самі популярні технології серед ваших користувачів, для встановлення списку яких необхідно попереднє дослідження.

2. Інсталюючи веб-додаток, звертайте увагу на те, чи дійсно додаток встановлюється у вказану вами директорію, базу даних, використовує обраний вами префікс та дотримується інших конфігураційних моментів.

3. Переконайтесь, що додаток можна встановити як localhost, так і віддалено.

4. Якщо процес інсталяції є ітерактивний, і кожний вибір користувача на певному кроці впливає на наступні дії, то необхідно буде пройти всі гілки, так як саме інсталяція є першим кроком користувача при роботі з вашим додатком.

5. Не забувайте про негативні тести: перевірка в умовах нестачі ресурсів (місця на дискові, оперативної пам'яті) чи привілегій, переривання процесу встановлення під час активної його фази (наприклад, надсилаючи сервер на перезаванатаження).

*Тестування навантаження*

Тестування навантаження імітує роботу з додатком певної кількості користувачів. Цей вид тестування здійснюється за допомогою спеціальних інструментів (наприклад jMeter), головна ціль яких – визначити профілі навантажень та штучно створити для них навантаження, які визначають граничні можливості додатку (чи сервера) в умовах роботи з ним того чи іншого користувача.

Отримана інформація підлягає детальному аналізу з наступним виявленням «вузьких місць» та граничних програмних чи апаратних можливостей, котрі в подальшому використовуються для забезпечення стабільності веб-сервера та самого додатку, що на ньому працює. Наведемо приклад з практики функціонування великого комерційного продукту, котрий довший час працював з різноманітними типами договорів. Одного разу в реліз випустили черговий особливо очікуваний тип договору, і на наступний день система повністю перестала працювати, а служба підтримки була завалена величезною кількістю звернень. Всьому виною став прорахунок в тестуванні: команда перевіряла одночасну роботу з десятками тисяч договорів, але ніхто не зміг передбачити, що на практиці мова піде про сотні тисяч, а іноді і про мільйони договорів. Тестування навантаження дозволяє виявити потенційні проблеми такого характеру ще на етапі тестування.

5. Особливості архітектури: «під прицілом» база даних

Ще однією раніше розглянутою складовою веб-додатків є база даних, в якій додаток зберігає всю необхідну інформацію. Для того, щоб база даних служила гідним сховищем інформації для вашого додатку, при тестуванні необхідно звертати увагу на такі основні моменти:

1. Інформація, внесена через інтерфейс, повинна бути збережена в базі даних в незмінному (первісному) вигляді.

2. Збережена в базі даних інформація повинна відображатися в будь-якій частині програми однаково (якщо іншого не вимагає бізнес-логіка програми).

3. Назви таблиць і структура бази даних повинні відповідати проектній документації.

4. Потрібно стежити за тим, щоб запити не оброблялись надто довго, а кількість з'єднань було достатнім. Моніторинг стану бази даних - один із важливих моментів тестування.

5. Варто враховувати, що видалення запису з бази даних не завжди супроводжується повним видаленням сутності. Іноді використовується так зване «псевдовидалення», і потрібно перевірити, чи правильно воно виконується.

6. Порожню базу даних на тестовому стенді рекомендується або заповнювати згенерованими випадковими даними, або знімати дамп з продакшена і після обфускації (обфускація – це процес, в результаті якого код програми чи інших даних набуває вигляду, важкого для аналізу) даних «заливати» їх в тестовану базу даних. Іноді кваліфікація тестувальника (або інша причина) не дозволяє виконати цей процес самостійно: в такому випадку рекомендується звернутися за допомогою до фахівців інфраструктури або розробникам.

***Запити***

Всі складові веб-додатів повинні взаємодіяти між собою і відбувається це за допомогою НТТР. Без НТТР наша багатостороння система не функціонувала б взагалі, так як НТТР – це протокол передачі даних, який займає одне із основних місць в нашій архітектурі.

Взаємодія здійснюється через повідомлення: на надісланий запит від клієнта повинна прийти відповідь сервера. Класичний запит/відповідь складається з 3 складових:

- стартова стрічка;

- заголовок;

- текст повідомлення.

При роботі з відповідями спеціаліст по тестуванню в першу чергу повинен звертати увагу на методи та коди станів, котрі присутні в стартовій стрічці.

Найпопулярнішими методами є GET, HEAD і POST:

1. Метод GET. Використовується для запиту вмісту, розміщеного на сервер

(наприклад, GET/path/resource?рaram1=value1¶m2 = value2 HTTP/1.1).

2. Метод HEAD. За своєю суттю не відрізняється від вищезгаданого методу, однак відповідь сервера на такий запит позбавлений «тіла», а практичне застосування орієнтоване на полегшене використання з метою отримання мінімальної інформації про сервер/продукт або його статус.

3. Метод POST. Даний метод використовується для передачі даних на сервер, однак його основа «ховається» в тіло, що відрізняє його від GET. Під час публікації цієї статті, наприклад весь текст поміщений в тіло POST-запиту; після обробки його сервером на сайт буде додано статтю.

Існують і інші методи: PUT, DELETE, CONNECT, TRACE, PATCH тощо. Те, що складові веб-додатку взаємодіють між собою за допомогою HTTP, є гарною новиною для фахівця з тестування, так як це дозволяє не просто відстежувати спілкування, вникаючи в логіку роботи і звіряючи її з технічними вимогами, але і впливати безпосередньо на додаток, прикладаючи свою руку до відправлених запитів.

Класичними додатками, які можна використовувати для генерації запитів, є Fiddler або Postman. Використовуючи Fiddler, можна з легкістю відстежувати всі запити від клієнта і відповіді, переглядати їх деталі, а також вносити свої зміни і відправляти модифіковані запити на сервер, оцінюючи поведінку системи в такому випадку.

На що звертати увагу при запитах?

1. Чи правильний метод використовується для того чи іншого запиту? Якщо ви відправляєте повідомлення, а на сервер йде тільки GET-запит, то щось тут явно не так.

2. Здавалося б, клік по одній і тій же функціональній клавіші генерує один і той же запит. Але є прецеденти, коли клік по кнопці приводив до ситуації, в якій JavaScript переписував запит кнопки, розміщеної поруч, таким чином змінюючи її призначення.

3. Вникайте у запити, які надсилаються. У них досить легко розібратися, особливо якщо це GET - вони логічно зрозумілі навіть пересічному користувачеві. Аналіз запитів - це можливість виявити прихований дефект набагато швидше, ніж здійснюючи його пошук в інтерфейсі.

4. Моніторте трафік на предмет запитів на інші (не ваші) сервера. Приклад з життя: фронтенд сайту робив фріланс-розробник, по завершенню роботи якого взялася за тестування фірма. Тестувальник мав звичку моніторити весь трафік тестованої програми: це дозволило виявити, що вищезгаданий розробник безсоромно сховав у «фронт» сайту запити, які працювали на благо його особистого інтернет-магазину.

5. Моніторте трафік, уважно стежте за кодами станів. Трішки докладніше зупинимося на цьому пункті.

***Коди станів очима адепта якост***і

Кожна відповідь на будь-який запит несе в собі масу корисної для розробників і фахівців з тестування програмного забезпечення інформації. Одним з джерел такої інформації може стати код стану: за цим кодом клієнт дізнається про результати його запиту і визначає, які дії йому робити далі. Набір кодів стану є стандартом, він описаний у відповідних документах RFC. Кожен раз, коли ви створюєте баг-репорт про непрацюючі посилання, інші елементи веб-сторінки або системні помилки, обов'язково вказуйте відповіді сервера: вони заощадять час розробнику при визначенні причин дефекту.

Всі коди можна поділити на групи (соті, двохсоті, трьохсот, чотирьохсоті і п'ятисоті) кожна група-«сотня» несе свій тип інформації (див.табл.1).

Таблиця 1. – Коди станів

| Код | Тип інформації коду | Призначення |
| --- | --- | --- |
| 1хх | Інформаційні | Інформують про передачу повідомлення. Самі повідомлення від сервера містять тільки стартову стрічку відповіді і, якщо необхідно, декілька специфічних для відповіді полів заголовку |
| 2хх | Повідомляючі про успішні операції | Інформування про успішну обробку запиту клієнта. Самим класичним та поширеним є «200 ОК» |
| 3хх | Перенаправляючі | Повідомляє, що для успішного виконання операції необхідно зробити інший запит. З даного типу п’ять кодів: 301,302,303,305 та 307 відносяться безпосередньо до перенаправлення. Адреса, за якою клієнту варто проводити запит, сервер вказує в заголовку Location |
| 4хх | Помилка на стороні клієнта (запит) | Вказує на помилки на стороні клієнта. При використані всіх методів, крім HEAD,сервер повертає пояснення причини |
| 5хх | Помилка на стороні сервера (відповідь) | Говорить про помилки виконання операції за виною веб-сервера |
|  |  |  |
|  |  |  |

На практиці, використовуючи при тестуванні спеціальні додатки (той самий Fiddler), ви без зусиль зможете впорядкувати свої запити і відповіді за кодом стану і відібрати, наприклад, всі 400-і і 500-і з подальшим їх аналізом. Таким чином дуже швидко «відловлюються» дефекти з «відваленими» стилями, скриптами, файлами, функціями додатку тощо.

**Відмінність веб-додатку від десктопного Багатокористувацька сутність веб-додаткі**в.

Широта аудиторії додатків накладає свій відбиток на специфіку роботи.

1. Один додаток одночасно може використовуватися величезною кількістю людей. Ми вже розглядали питання навантажувального тестування, але також слід звернути увагу на те, що в число користувачів можуть входити представники різних культур, мов і релігій. Нам необхідно пам'ятати про це, особливо якщо мова йде про тестування міжнародного додатка.

2. Кожен користувач може мати свої рівні доступу. В ідеальному варіанті тестувальник створює для себе матрицю рівнів доступу і тестує кожен доступ окремо (див.рис. 3).

3. Користувачі з одним рівнем доступу можуть звертатися до одних і тих же сутностей, що призводить до конкурентного доступу. Тестується це досить просто. Для прикладу розглянемо систему, що має справу з договорами, які можна створювати, публікувати, редагувати, анулювати. Алгоритм роботи такий: під декількома вікнами в режимі інкогніто авторизуємося в додатку під користувачами з різними рівнями доступу; далі обрану для тесту сутність відкриваємо на редагування, а під другим обліковим записом цю ж сутність пробуємо перевести в статус «Анульовано» - на цьому етапі повинен спрацювати контроль на конкурентний доступ. Операція анулювання блокується, а користувачу видається повідомлення про те, що сутність редагується іншим користувачем (поведінка і пріоритет дій визначається відповідно до вимог і особливостей продукту, але логіка не змінюється).

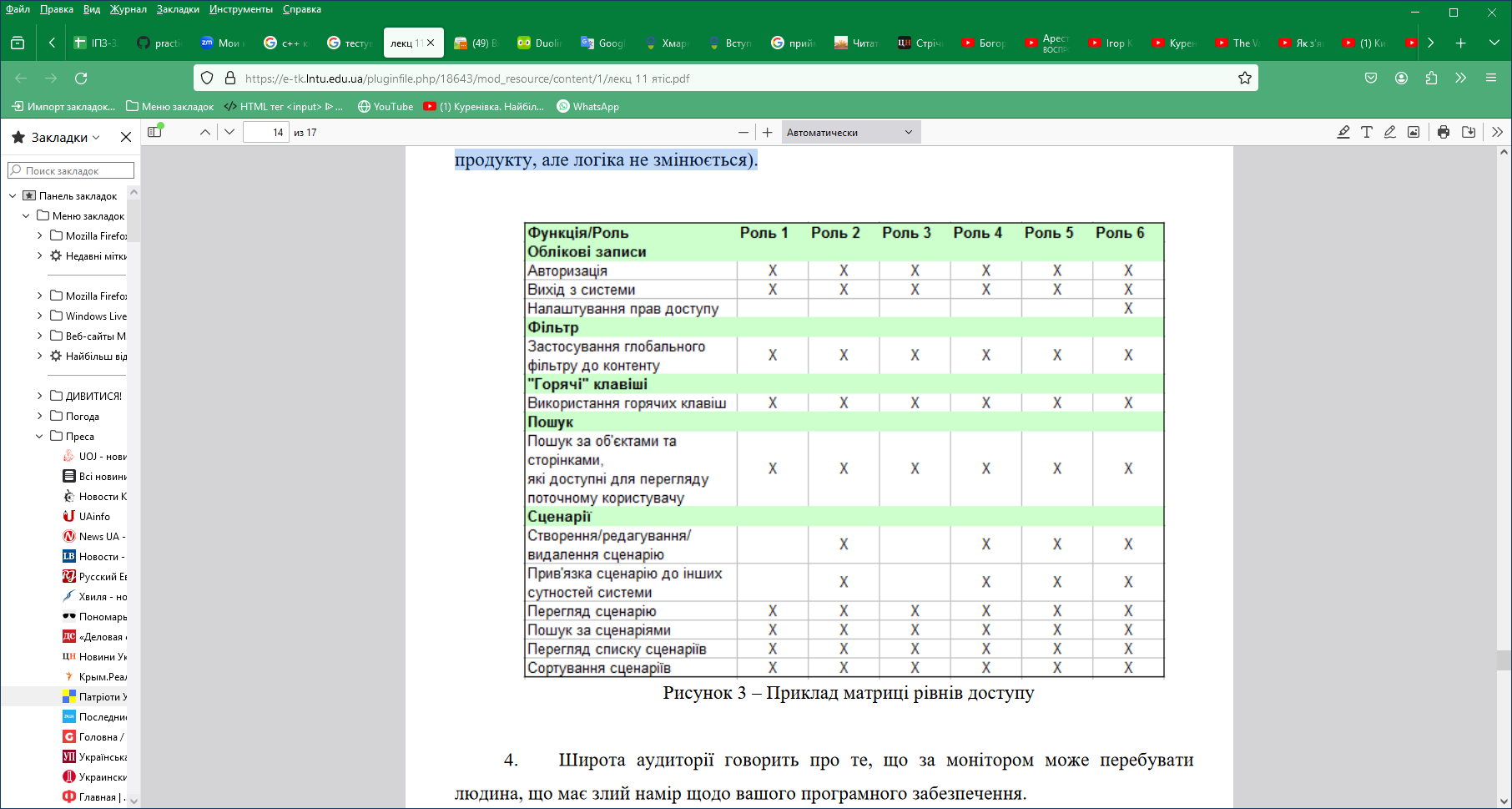


Рисунок 3 – Приклад матриці рівнів доступу

4. Широта аудиторії говорить про те, що за монітором може перебувати юдина, що має злий намір щодо вашого програмного забезпечення. Мережеві пристрасті: веб-додаток в різних умовах передачі даних Веб-додатки активно використовують мережу, і це є джерелом можливих проблем. Такою, наприклад, є використання додатку в умовах низької швидкості передачі даних (в браузер Google Chrome, наприклад, вбудована функція Throttling, яка дозволяє сильно занижувати швидкість передачі даних), в умовах втрати пакетів або при відключенні мережі під час активної фази роботи програми (спосіб імітації: спочатку робимо швидкість передачі даних за допомогою Throttling мінімальною, а потім перериваємо мережеве з'єднання під час обробки запиту).

У будь-якому з описаних вище випадків програма має працювати коректно. При «падінні» запиту (time out) або іншої проблеми ми повинні, перезавантаживши сторінку, знову отримати повністю працюючий веб-додаток без будь-якого натяку на щойно пережиту «шкоду». Чи для всіх функцій програми необхідні подібні тести? Ні в якому разі! В майбутньому можете орієнтуватися на свій досвід, а на перших етапах в цих питаннях краще проконсультуватися з розробниками.

*Тестування безпеки веб-додатків*

Тестування безпеки - окремий напрямок тестування, яке вимагає від фахівця фундаментальних знань технічного характеру і хорошої профільної кваліфікації. Зазначимо ряд загальних моментів, які можуть допомогти будь- якому тестувальнику знаходити класичні уразливості, не допускаючи їх вихід на продакшен. Питання безпеки додатків регламентуються OWASP Guide, CHECK, ISACA, NIST Guideline, OSSTMM.

Існує ряд принципів безпеки, до яких відносяться конфіденційність, цілісність і доступність:

1) Конфіденційність - обмеження доступу до тієї чи іншої інформації для певної категорії користувачів (або навпаки надання доступу лише обмеженій категорії).

2) Цілісність включає в себе:

а) можливість відновити дані в повному обсязі при їх пошкодженні;

б) доступ на зміну інформації тільки певної категорії користувачів.

3) Доступність - ієрархія рівнів доступу і чітке їх дотримання.

Перерахуємо класичні уразливості сучасних веб-додатків:

1. XSS - генерація на сторінці продукту скриптів, які становлять небезпеку для користувачів продукту;

2. XSRF - вразливість, при якій користувач переходить з довіреної сторінки на шкідливу, де крадуться цінні для користувача дані;

3. Сode injection (PHP, SQL) - ін'єкція частини виконавчого коду, яка робить можливим отримати несанкціонований доступ до програмного коду або бази даних і вносити в них зміни;

4. authorization bypass - це вид уразливості, при якому можна отримати несанкціонований доступ до облікового запису або документів іншого користувача;

5. переповнення буфера - явище, якого можна досягти у шкідливих цілях, по своїй суті являє використання місця для запису даних далеко за межами виділеного буфера пам'яті.

При тестуванні рекомендовано використовувати чіт-листи вразливостей XSS Filter Evasion Cheat Sheet і MySQL SQL Injection Cheat Sheet.

З чого починати тестувати?

Закцентувати вашу увагу на необхідності узгодити всі ключові аспекти з відповідальними особами (аналітиками, проектними менеджерами, розробниками) ще до початку тестування. Необхідно заздалегідь з'ясувати наступні моменти:

* мета тестування (з їхньої точки зору);
* види тестування, які необхідно провести;
* специфіку програми та її цільову аудиторію;
* перелік пристроїв, на яких необхідно проводити тестування; список
* операційних систем і браузерів для тестування;
* роздільну здатність екранів на яких необхідно перевірити додаток;
* чи передбачені вимоги до різного роду форм, чи є стайл-гайд, актуальне технічне завдання;
* необхідність надання конкретної документації за результатами тестування (звіт, чек-листи, тест-кейси тощо).

Отримавши відповіді на ці питання, про які часто забувають новачки, ви заощадите свій час. Далі переходите до безпосередньої підготовки оточення і формування стратегії тестування.

Ніколи думати, потрібно тестувати!

Будь-яке тестування вимагає змістовного підходу із застосуванням технік тест-аналізу і тест-дизайну. В іншому випадку ви ризикуєте назавжди залишитися «monkey-тестером», цінність праці якого буде мізерна. В цілому, ключові положення тест-аналізу і тест-дизайну застосовні як до тестування десктоп-додатків, так і до веб, але з суттєвим застереженням: ви повинні враховувати всі згадані в статті нюанси. Окремо закцентувати увагу на тому, що без стійкого розуміння методик і способів застосування тест-аналізу і тест- дизайну тестувати якісно програмне забезпечення практично неможливо Розглянемо класичний набір методик тест-дизайну, які можна застосовувати при тестуванні веб-додатків:

* розбиття на класи еквівалентності; попарне тестування;
* тестування переходів станів; аналіз граничних значень;
* тестування за допомогою таблиць рішень; методика тест-турів Віттакер і безліч інших.

[Функціональне тестування](https://artjoker.ua/uslugi/qa_os_testirovanie/) системного додатку, наприклад, мережі компанії відрізняється від тестування мобільного застосунку, який охоплює тисячі, якщо не мільйони, клієнтів. Тестування мобільних додатків передбачає тестування багатьох конфігурацій, а отже, займає кілька етапів.

## **Розмежування між тестуванням мобільних і веб-додатків**

* 1. **Одна з найочевидніших відмінностей - мобільні додатки призначені для більшої кількості користувачів, більш різноманітних пристроїв, ніж веб-додатки.**

Хоча відмінності між цими двома категоріями комп'ютерів зменшуються в міру того, як дедалі більше веб-додатків стають доступними в мобільних форматах, складність мобільних додатків постійно розширюється зі збільшенням складності функцій, які мобільні пристрої повинні підтримувати. Мобільний додаток може працювати на декількох пристроях, включно зі смартфоном, планшетом, кардіостимулятором, системою блокування або фітнес-трекером.

* 1. **Веб-браузери були створені стаціонарно.**

У той час як мобільні додатки повинні постійно виконувати "на ходу" обчислювальні функції, веб-додатки зазвичай зосереджені на стаціонарних портативних або настільних операціях, з класичними функціональними можливостями миші, курсору або маршрутизатора Wi-Fi, які менш задіяні в мобільних додатках. Коли тестування web-додатків переходить до тестування мобільних додатків завдання послідовно розширюється. Тестування мобільних додатків має стосуватися кількох різних функцій щодо кількох різних середовищ у кількох різних умовах.

Збільшення інтересу користувачів та їхньої прив'язаності до мобільних пристроїв призвело до того, що QA повинні перевіряти застосунки на:

* **постійний зв'язок**
* **управління повідомленнями**
* **синхронізацію на кількох платформах**

Більш того, важливіше, ніж цілі, завдання, планування - клієнт. Таким чином, тестування програмного забезпечення являє собою завдання координації та забезпечення рівня функціональності мобільних пристроїв, яка буде враховувати вимоги та переваги клієнтів.

## **Технічні виклики**

### **Відмінності між мобільним і веб-тестуванням:**

* 1. **RAM і обмеження зберігання.**

Багато мобільних пристроїв, як і раніше, постачаються з 1 або 2 ГБ оперативної пам'яті, а також із порівняно невеликими 16 ГБ SSD. Це створює серйозні обмеження для оперативної пам'яті та ємності сховища для тестування, особливо щодо величезного обсягу пам'яті та сховища, які доступні будь-якому сучасному веб-браузеру. Крім того, такі послуги, як рекламні платформи, можуть серйозно сповільнити роботу мобільного браузера, тож перенесення вашого веб-додатка на телефон або планшет може викликати труднощі.

* 1. **Різні взаємодії для різних користувачів.**

Взаємодії для комп'ютерів і ноутбуків стабілізовано понад 30 років - миша та клавіатура - це стандарт для всього, від роботи у Facebook до флеш-ігри. Це становить особливості тестування веб додатків.

Проблема з боку мобільних додатків полягає в широкому спектрі сенсорних дій - прокрутка, витягування, защемлення + голосові можливості (Siri та Google Now). Специфічні для конкретного пристрою інновації, як-от жести hand wave на деяких гарнітурах Samsung або новий набір аудіо iPhone, додають складнощів на тестування ios додатків та Android-додатків.

* 1. **Індивідуальні типи додатків.**

Десктопний веб-додаток розроблено на HTML, CSS і JavaScript з деякими варіантами залежно від того, які платформи розробник хоче використовувати. Мобільні додатки не такі прості. Вони можуть бути створені, як нативні додатки на Java або Objective-C, або як гібридні, які можуть використовувати спеціальні платформи для представлення системних API в якості API-інтерфейсів JavaScript, адресованих веб-кодом. Дуже важливо, щоб було розроблено roadmap для платформи, щоб керувати випробуваннями для всіх типів тестування.

* 1. **Поведінка з підключенням і без нього.**

Веб-додаток, що працює на Mac або ПК, буде марним без підключення до Інтернету. Те ж саме для мобільного веб-додатка. Однак нативний або гібридний мобільний додаток може мати автономні функції. Деякі ігри можуть відтворюватися без підключення, а редактори документів також можуть зберігати функціональність у режимі польоту. Необхідно перевіряти, як мобільні додатки поводяться при перемиканні між з'єднаннями або використанні нестабільних мереж.

Це допоможе провести тестування андроїд-додатків, а також iOS і веб-додатків якісніше.

Звісно, існує певний збіг між веб-і мобільними додатками. Багато мобільних додатків частково відносяться до веб-додатків, і тестувальники дотримуються деяких загальних схем тестування в рамках гнучкого циклу розробки. Наявність програмного забезпечення для управління тестуванням є ключовим будівельним блоком, оскільки ви формуєте загальну стратегію тестування і звертаєте увагу на врахування широкого кола робочих процесів, ресурсів і можливих проблем, пов'язаних як з мобільними, так і з веб-додатками.

### **Види тестувань**

Важливо зауважити, що ручне тестування, як і автоматизоване, відіграють важливу роль для QA. Машина може значно заощадити час і полегшити процес, але не всі тестові процеси обійдуться без участі людини.

Автоматизоване тестування використовується для:

* тестування модулів і бібліотек
* відповідності UI/UX
* API

Не забуваємо і про тестування безпеки веб-додатків, це забезпечує комфортну роботу для клієнтів і позбавляє від обробки багів після релізу додатків.

Особливості тестування мобільних додатків - ключовий фактор для того, щоб отримати чудовий продукт. Тестувальники команди Artjoker уважно ставляться до будь-яких дрібниць і розуміють важливість правильного функціонування як web, так і мобільних додатків. Ми допоможемо провести тестування додатків android або iOS на високому рівні.

**Поняття багатомовності та локалі**

**Мультимовні додатки** – це програми, які підтримують дві і більше мови інтерфейсу.

Зараз, в еру глобалізації, майже все перекладається на кілька мов – плакати, банери, дорожні знаки, покажчики в музеях та ін. І хоча, здавалося б, майже кожен сьогодні володіє англійською, але часто буває, що людині доступна тільки рідна мова.

У сфері IT мультимовними можуть бути сайти, ігри, додатки для мобільних пристроїв і ПК. Тестування правильності перекладу і мовних аспектів називається **тестуванням локалізації.**

Не тільки переклад тексту важливий в багатомовних додатках, а й локаль. Розглянемо це поняття.

**Локаль *(от англ. locale)*** – це регіональний стандарт, набір налаштувань інтерфейсу користувача. До локалі відносяться: мова програми, відображення дати і часу, часових поясів, одиниць виміру, порядок написання прізвища, імені та інші особливості.

Таким чином, тестування багатомовних додатків вже стає чимось більшим, ніж просто звірка слів на різних мовах.

**Тестування багатомовних додатків**

Якщо при тестуванні багатомовних додатків брати до уваги і перевірку локалі, то пунктів для тестування виходить дуже багато. Розглянемо найчастіші перевірки, які варто проводити в цьому випадку.

**1. Підтримка мови**

Перш ніж тестувати сам переклад, необхідно перевірити чи підтримуються різні мови додатком. Визначити це можна кількома способами: за IP-адресою, геолокацією, обраним регіоном в девайсі, за адресою сайту (для веб-додатків), обиратися в опціях програми і т. ін. Додаток має враховувати всі ці випадки та правильно обирати мову відображення.

Помилок може бути декілька:

* залишається мова додатку за замовчуванням (зазвичай англійська) і вона не змінюється на обрану після натискання на неї;
* обрана мова відображається у вигляді квадратів, незрозумілих символів або всі елементи взагалі порожні – це помилка в кодуванні;
* замість обраної мови текст перекладається на будь-яку іншу і, яку мову не обрати, потрібної просто ніде немає.

На сьогоднішній день більшість додатків мають кілька мов для перекладу. Виходячи з кількості носіїв та їх популярності, це англійська, китайська, іспанська, французька, російська, португальська, німецька, арабська, японська та корейська мови.

Компанії-розробники намагаються орієнтуватися на цю інформацію і не втрачати потенційних клієнтів, з огляду на їх кількість.

**2. Правильність перекладу**

Частою проблемою є те, що перекладачі, які погано знають мову, вибирають інше значення слова і шокують користувача, змушуючи його самостійно додумувати значення кнопки. Наприклад, слово «follow», крім «підписатися», має кілька інших значень, тому на сайті можуть зустрітися фрази на кшталт «Ідіть за нами» або «Супроводжуйте нас».

Але важливо не тільки правильно перекласти речення, а ще й адаптувати його сенс до конкретної мови та національності. Багато проблем розробникам доставляє наявність різного роду в деяких мовах. Їм доводиться дублювати фразу в коді в залежності від статі користувача, а також враховувати, що не тільки іменники мають свій рід, а прикметники та дієслова також по-різному схиляються.

Бувають випадки, коли вибравши не англійську мову в додатку, поряд зі словами вибраної мови все одно зустрічаються неперекладені елементи. Ще гірше, коли Ви бачите декілька мов і виходить своєрідний Вавілон, де все змішалося і користувач не може розібратися в додатку. У такому випадку компанія точно втратить свого клієнта.

**3. Відповідність розміру тексту перекладу**

Не варто забувати про те, що одне і те ж слово на різних мовах має різну довжину. Відповідно, в додатку повинна або змінюватися довжина елемента, або розмір шрифту, інакше текст буде виходити за межі кнопки, обрізатися або накладатися на сусідній текст. При цьому зміна шрифту не повинна позначатися на читабельності фрази. Краще пожертвувати точним змістом та адаптувати текст по-іншому, ніж залишити точний переклад, але користувачі не зможуть розібрати, що написано.

**4. Переклад статичних картинок**

У деяких додатках, є банери або картинки, на яких теж присутні написи. Тому програми повинні містити дублікати цих зображень, але з текстом на підтримуваних мовах.

**5. Напрямок тексту**

Цей пункт відноситься до тих мов, які пишуться не звичним для нас способом зліва направо, а справа наліво. Це, наприклад, арабська та іврит. Тестувальникам потрібно враховувати цей фактор, так як для жителів цих країн важливе подібне написання тексту.

**6. Відповідні посилання**

Якщо в додатку обрати португальську мову, це означає, що не можна відправити користувача на російську версію сторінки, припустимо, Вікіпедії. Це ж стосується і посилань на інші ресурси, які вбудовані самою компанією-розробником, а не рекламою.

**7. Стандарти відображення імен**

Варто не забувати про порядок відображення прізвища та імені на різних мовах. В українській мові прийнято вказувати першим прізвище, а потім ім'я (Шевченко Марія), а в англійській мові все навпаки (Maria Shevchenko).

Також це стосується кількості символів для імені. Наприклад, в Південній Кореї повне ім'я складається в більшості своїй з трьох складів, тому на багатьох сайтах і в додатках форми реєстрації обмежують введення імені до трьох символів.

**8. Стандарти відображення валют**

Навіть валюти в різних країнах відображаються по-різному. Положення даного символу може бути перед або після числового значення, а десятковий і тисячний роздільник може відображатися у вигляді крапки, коми або ж пробілу в залежності від країни.

Нижче для прикладу наведено таблицю відображення валют з різних країн.

|  |  |
| --- | --- |
| **Країна** | **Відображення** |
| Україна | 200 000,50 ₴ |
| США | ₴ 200,000.50 |
| Ізраїль | ₴ 200,000.50 |
| Германія | 200.000,50 ₴ |
| Китай | 200000.50₴ |

**9. Стандарти відображення дат і часу**

Також зі стандартів локалі варто звернути увагу на те, чи в правильному форматі відображається дата і час країни, мову якої було обрано.

У даті, крім порядку дня, місяця та року, також відрізняються і роздільники між ними. Існує більше 10 форматів запису дати, але тут буде представлено 5 найбільш часто використовуваних.

|  |  |  |
| --- | --- | --- |
| **Формат** | **Приклад дати** | **Приклади країн** |
| дд.мм.рррр | 01.02.2019 | Україна, Німеччина, Норвегія, Росія |
| м/д/рррр | 2/1/2019 | США |
| дд/мм/рррр | 01/02/2019 | Великобританія, Іспанія, Франція, Італія, В'єтнам |
| рррр-м-д | 2019-2-1 | Китай |
| рррр-мм-дд | 2019-02-01 | Швеція, Канада, Польша, Литва |

Що стосується формату часу, то майже в усьому світі прийнято французький 24-годинний стандарт. Однак в таких країнах як США, Канада, Австралія і Нова Зеландія користуються англійською 12-годинною системою.

**10. Стандарти відображення розділових знаків**

Всім відомо, що в українській і російській мовах пробіл ставиться після коми, а не до, знак оклику в реченні тільки один – в кінці, а між деякими складеними словами є дефіс. Однак, в кожній мові є свої правила, які стосуються розділових знаків та їх відображення. Нижче представлені скріншоти особливостей пунктуації в деяких мовах.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| UA | EN | FR | ES | AR |
| Ви впевнені? | Are you sure? | Êtes-vous sûr ? | ¿Está seguro? | هل أنت متأكد؟ |

Лапки існують у всіх мовах, але виглядають, як і національні костюми, по-різному.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Мова** | **«…»** | **„…“** | **“…”** | **‘…’** | **《…》** | **「…」** |
| Українська | Стандарт, крапки і коми зовні | Альтернативні у фразі всередині лапок і при написанні від руки | Альтернативні у написанні від руки | - | - | - |
| Англійська | - | - | Стандарт при цитуванні, крапки і коми всередині | Стандарт для заголовків | - | - |
| Німецька | - | Стандарт | - | - | - |  |
| Французська | Стандарт, нерозривний пробіл перед лапками | - | - | - | - | - |
| Іспанська, португальська, бразильська, італійська, арабська | - | - | Стандарт | - | - | - |
| Китайська |  | Тільки в перекладах або в сусідстві з фразами з європейських мов | Тільки в перекладах або в сусідстві з фразами з європейських мов | Тільки в перекладах або в сусідстві з фразами з європейських мов | Стандарт для назв книг, фільмів і інших авторських творів | Стандарт для інших випадків |

**11. Конвертація міри та грошових одиниць**

Майже всі країни користуються однаковими стандартами міри – кілометрами, метрами, літрами, кілограмами, – проте, наприклад, в США зовсім інша система, люди там використовують милі, галони, фути і дюйми. Тому іноді в додатках можна вибрати англійську для конкретної країни, при цьому всі одиниці виміру і їх значення повинні перекладатися для конкретного регіону.

Те ж стосується і грошових одиниць. Якщо це інтернет-магазин, то в додатку повинен бути підключений конвертер валют, який буде змінювати не тільки відображення валюти, як було описано у восьмому пункті, а й конвертувати грошові одиниці згідно з курсом.

При тестуванні багатомовних додатків потрібно виконувати багато різних перевірок. У цій статті були названі основні, але і це не все, що можна застосувати. Важливо пам'ятати, що в світі багато мов і країн, у кожної є свої особливості та культурні аспекти. Тестування локалізації – це важка і копітка робота, яка займає багато часу у тестувальників, але для компанії-розробника мультимовні додатки – це відмінний шанс вийти на світовий ринок та отримати визнання у великої кількості користувачів.

# Тестування ігор: типи та способи тестування мобільних/настільних програм

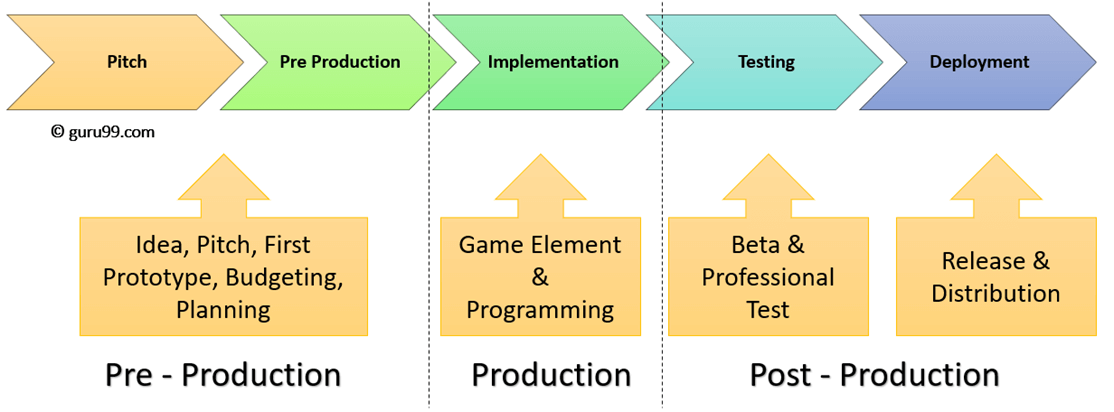
## Що таке тестування гри?

**Тестування гри** це процес тестування програмного забезпечення для тестування відеоігор для контролю якості. Основна мета тестування ігор — визначити та виявити дефекти та помилки у відеогрі та покращити стабільність і продуктивність. Тестування ігор — це компонент розробки ігор, який допомагає забезпечити розгортання відеоігриyed не містить помилок.

## Життєвий цикл розробки гри

**Попереднє виробництво:** На цьому етапі готується ідея гри, розкадрування, характеристики, аналіз вимог і документація. Ця фаза включає документацію технічного дизайну та специфікації функцій, гру архітектура, накладання кадрів, анімація. Послідовно розглядаються предмети

* Музика, Камера (zoom вхід і вихід, повтор, кінематографічний вигляд) атрибути гравця та дії
* Логіка ходу гри, правила та умови для досягнення наступного рівня
* Тригери об’єктів і подій, результати, рух і позиціонування гравців, статистика гравців,
* Неінтерактивна послідовність, спеціальні ефекти, титульні екрани, дії з кількома кнопками
* Геймпад, відеокліп, ефекти удару/вібрації, юридичні тексти, використання функцій кнопок, використання аналогового та цифрового режимів.

[](https://www.guru99.com/images/1/070918_0431_GameTesting1.png)Життєвий цикл розробки гри

**Production** : на цьому етапі виконується фактичне кодування. Цей етап включає кодування, інтеграцію різних модулів.

**Тестування та розгортання**: На цьому етапі виконуються функціональні тести, регресійне тестування, альфа, бета, золото. Тестування покриття та потоків, цілісності даних, тестування на основі алгоритму, тестування шляху, інкрементальне тестування також виконується за допомогою інструментів тестування мобільних ігор.

## Чим тестування ігор відрізняється від іншого тестування програмного забезпечення.

Тестування ігор - це a **повторюваний процес** кожна нова збірка може містити помилки і повинна бути ретельно перевірена.

Усе тестування гри дотримується базової структури, незалежно від розміру гри та часу, необхідного для створення гри.

Професіонал із забезпечення якості має вивчити правила гри та вимоги. Зрозумійте загальну складову гри archiтектура та файл archiтектура, потік, файлові структури та залежності, пов’язані з грою. З кожним новим прототипом гри потрібно часто переглядати тестові документи, щоб оновити будь-які зміни в специфікаціях, нові тестові випадки тестування гри та нову підтримку конфігурації. Тестер відеоігор повинен переконатися, що не було внесено нових проблем.

Робота тестувальника ігор передбачає:

* Класифікуйте вимоги, виходячи з цільової мети та цільової аудиторії.
* Визначте вимоги користувача та системи, і їх слід класифікувати на функціональні, нефункціональні вимоги, вимоги домену
* Визначте предмети, що підлягають перевірці, елементи, що не підлягають тестуванню, цілі та заходи щодо функціональних і нефункціональних вимог
* Перевірте, чи функціональні вимоги повні, послідовні та зрозумілі
* Визначте настроювані вимоги, суперечливі вимоги
* Визначення взаємозалежних вимог є одним із завдань тестування гри.
* Пріоритезація вимог на основі унікальності, комplexність, критичність
* Визначте тему гри, персонажів, анімації, штучного інтелекту, кінематографії, перегляду камери, ігрового процесу

## Види тестування гри

Нижче наведено популярні методи тестування ігор:

### 1) Функціональне тестування

Тестери перевірки функціональності шукають загальні проблеми в грі або її користувальницькому інтерфейсі та графіці, як-от проблеми з ігровою механікою, проблеми зі стабільністю та цілісністю ігрових ресурсів. Тестування інтерфейсу користувача забезпечує зручність гри

**Приклад:** Перевірка кольорів і фону, структури меню, орієнтації екрана та роздільної здатності екрана, розміру шрифту, помилок вирівнювання, зручності використання, системної навігації, як-от час завантаження, тайм-аут і відображення, сортування, повідомлення підтвердження, послідовності, анімація та аудіоелементи, аспекти гри, інструкції , і діалогові повідомлення. Взаємодія з користувачем, інтерфейс користувача, тестування транзакцій, калібрування та тестування точності камер мобільного телефону, роздільна здатність екрану, тестування адаптивного мобільного дизайну, тестування якості звуку

### 2) Тестування на сумісність

Перевірка сумісності гри з різними пристроями та різними конфігураціями апаратного та програмного забезпечення.

**Приклад**: установіть і видаліть гру на всіх підтримуваних консолях/настільних комп’ютерах/мобільних телефонах.

### 3) Тестування продуктивності

Перевіряється загальна продуктивність гри.Налаштування продуктивності виконується для оптимізації швидкості гри.

**Параметри важливості, перевірені під час тестування продуктивності**

* Час відповіді на клієнті та серверах, час завершення транзакцій, продуктивність при піковому навантаженні, довговічність, покриття мережі, витік пам’яті, низький рівень пам’яті, низький заряд акумулятора, час, витрачений на завантаження програм, одночасний доступ (кілька користувачів) доступ до сервера програми, швидкість, пропускна здатність, надійність, масштабованість тощо.
* **Споживання батареї та продуктивність графіки:** Виміряйте споживання батареї мобільної гри. Споживання батареї має бути оптимальним протягом тривалого часу, і реакція гри має бути задовільною за різних великих навантажень на різних пристроях
* **Обмеження процесора та пам'яті**: Лічильники продуктивності використовуються для вимірювання споживання процесора та пам’яті програмою.
* **Мережеве підключення**: Вимірює час відгуку мобільних ігор у різних типах мереж (Wi-Fi, 2G, 3G, 4G). Це дає загальне уявлення про те, наскільки добре гра працюватиме в ненадійних мережах. Він також перевіряє з’єднання між мобільними пристроями, центрами обробки даних або хмарою. Відстежуються всі часи піку, нестабільні з’єднання, дублювання даних, втрата пакетів, фрагментація даних.
* Тестування продуктивності мобільних ігор, особливо MMO

### 4) Відповідність/тестування відповідності

Відповідність інструкціям Marketplace (наприклад, політики Apple App Store), відповідність корпоративній політиці (наприклад, заборонений вміст. Відповідність також може стосуватися регуляторних органів, таких як PEGI та ESRB. Гра націлена на певний рейтинг вмісту. Якщо є неприйнятний вміст, який невідповідні бажаному рейтингу, тоді вони ідентифікуються та повідомляються. Навіть одне порушення під час подання на затвердження ліцензії може призвести до відхилення гри, що спричинить додаткові витрати на подальше тестування та повторне подання.

**Приклад**: Якщо гру планується опублікувати в європейських країнах, перевірте на конвертацію PAL, якщо гру створено для Північної Америки, перевірте на конвертацію NTSC.

### 5) Тестування локалізації

Тестування локалізації стає надзвичайно важливим, коли гра орієнтована на глобальні ринки. Назви ігор, вміст і тексти потрібно перекладати та тестувати на пристроях кількома мовами. Ці типи тестів можна виконати швидко (за допомогою хмарного доступу до пристрою та автоматизації тестування).

**Приклад**: потреби локалізації в регіоні MENA (Близький Схід/Північна Африка), арабська локалізація (підтримка тексту справа наліво, двонаправлені дисплеї), тестування псевдолокалізації, double-байтові символи (для мов Східної Азії), місцевий час/дата, валюта, формат адреси та інші місцеві вимоги.

### 6) Випробування на замочування

Це автоматизоване тестування гри передбачає залишення гри на тривалий час у різних режимах oперації. Наприклад, режим холостого ходу призупинено або на екрані заголовка. Вимочування може виявити витік пам'яті або помилки округлення.

**Приклад**: Гра почалася, і персонаж змушений стояти. Ця техніка використовується для виявлення збоїв, спричинених витоком пам’яті та іншими несправностями ігрового движка.

### 7) Тестування відновлення

У програмному забезпеченні, тестування відновлення перевіряє, наскільки добре програму можна відновити після збоїв, апаратних збоїв та інших подібних збоїв. Програма вимушено виходить з ладу, і пізніше буде спостерігатися, як він відновлюється після умов відмови та навколишнього середовища.

**приклад:** Під час роботи ігрової програми раптово перезапустить ігрову консоль і перевірте цілісність даних

### 8) Тестування безпеки

Це робиться для того, щоб перевірити, наскільки захищено програмне забезпечення від зовнішніх загроз. Захист даних від зовнішніх загроз, неконтрольованих обмежень доступу до системи, порушення даних, помилки системи зв'язку та слабке шифрування алгоритмів.

**приклад:** Зміна URL-адреси з /login на /play на ігровому сайті не повинна дозволяти прямий доступ до ігор.

### 9) Тестування інших ігор

Тестування реальних або віртуальних персонажів. У багатокористувацьких відеоіграх підключення до сервера та синхронізація статусу гри – це дві критичні області, які потрібно перевірити.

**Приклад**: багатокористувацькі тривимірні гоночні ігри.

Тестування нових функцій, таких як оновлення статусу гри, запрошення друзів, обмін преміальними подарунками тощо. Це забезпечує багатий ігровий досвід для користувача.

**Приклад**: Facebook, блоги

**Тестування звуку**

Тестування помилок під час завантаження файлів, прослуховування звукових файлів на наявність помилок або спотворень, CC Profiler для аналізу кольорових коментарів

**База даних і статистика гри**

Перевірка бази даних за допомогою налагодження, щоб перевірити, чи гра правильно використовує дані. Переконайтеся, що дані завантажено в потрібному місці та містять правильну інформацію

### Білий-ящик Тестування

Тестування для ігор фокусується на архітектурі, інтеграційні та системні аспекти мобільної гри.

1. **Перевірка коду**: Переглядається вихідний код, аналізується логіка програми та типові помилки програмування, відповідність стандартам кодування.
2. **Тестування фокусу:** Фрагменти коду подаються до ізольованих модулів, а вихід аналізується.
3. **Аналіз даних:** Використання, інтерпретація та маніпулювання даними аналізуються та перевіряються для різних модулів.
4. **Тестування шляху та потоку:** Виконується правильна послідовність об'єктів.
5. **Тестування на основі алгоритму**: Тестування певного ігрового сценарію або функції шляхом встановлення змінних даних, значень даних у коді та виконання його в середовищі виконання.
6. **Аналіз штучного інтелекту:** Генерується статистика виконання програмованих рухів і ігор компонента ШІ. Результат перевіряється, щоб перевірити, чи використовуються всі програмовані рухи. Приклад: використовується бокове захоплення сноуборду та гра (комбінація ударів кулаком/ногою в різних напрямках).

## **Допоміжні ігри з використанням адаптивних технологій**

Допоміжні ігри також відомі як доступні ігри. Функції розроблено з використанням адаптивної технології для людей із різними вадами, як-от слабкий зір, розмитість зору, сліпота, нездатність розрізняти кольори, порушення мовлення, слуху, когнітивні функції, моторика та рухливість.

Cardinal Direction (CD), Tower of London (TOL) — це дві популярні ігри, модифіковані для користувачів із вадами зору. У цих іграх візуальні подразники замінюються звуковими вводами.

Тестер відеоігор повинен звернути увагу на наступне під час тестування такої гри

1. Кольори повинні мерехтіти у вигляді візерунка, а тони гратимуть для кожного кольору.
2. Кожен колір має супроводжуватися відповідним тоном.
3. Візуальні дані потрібно описувати словами, щоб люди з вадами зору не стикалися з проблемами отримання за допомогою програм зчитування з екрана.
4. Гравець повинен чути звуки в грі в трьох вимірах і повинен орієнтуватися в слові за допомогою 3D-аудіо на сенсорному екрані та просторового звуку

## Ігрові показники, які повинен знати тестувальник.

**DAU/MAU (активні користувачі за день/активні користувачі за місяць):** Співвідношення активних користувачів, які грають щодня, до кількості активних користувачів щомісяця. Також зазвичай називають фактором липкості.

**Сесія:** Кожен раз, коли користувач відкриває програму, це зараховується як сеанс. Тут основна увага приділяється середній кількості сеансів на DAU.

**Рейтинг завантажень:** Ранг гри в певному магазині програм (iOS, Android Грати) шляхом щомісячного завантаження гри.

**Збереження:** Дуже важливий показник для android тестер для безкоштовної гри. Щоб розрахувати утримання, розділіть користувачів на когорти на основі дня завантаження програми.

**Показники ефективності:** Це для відстеження продуктивності онлайн-ігор або постійних ігор. Частота кадрів, з якою гра виконується на апаратній платформі клієнта, або, у випадку з ігровим сервером, його показники стабільності та продуктивності можна використовувати для моніторингу змінних функцій і оновлень.

## Ключові ризики під час тестування гри

1. Гра не створює переконливих вражень для цільової аудиторії.
2. Гра не має дизайну, орієнтованого на гравця
3. В іграх відсутні фактор веселощів і захоплюючий геймплей.
4. Гра не унікальна, змагальна, швидка.
5. Гра виходить з ладу через технічні проблеми, несправні функції, критичні помилки, поганий звук музики та погане відео.
6. Вартість розробки гри перевищує бюджет
7. Гра повинна мати простий естетичний дизайн і геймплей.

## Підсумки

* Життєвий цикл розробки гри складається з трьох етапів: підготовка до виробництва, виробництво, тестування за допомогою інструменту тестування гри та розгортання
* Тестування ігор — це повторюваний процес, у кожній новій збірці можуть бути помилки, тому її потрібно ретельно перевіряти за допомогою інструментів автоматизації тестування ігор
* Різні типи тестування: 1) функціональне тестування,2) Перевірка сумісності, 3) Тестування продуктивності, 4) Відповідність/тестування відповідності, 5)Тестування локалізації, 6) Тестування затримки, 7) Тестування відновлення, 8) Тестування безпеки
* білий Box Тестування для ігор фокусується на archiтектурні, інтеграційні та системні аспекти мобільної гри, які включають 1) перевірку коду 2) тестування фокусу 3) аналіз даних 4) тестування шляху та потоку 5) тестування певного алгоритму 6) Штучний інтелект аналіз
* Допоміжні ігри також відомі як доступні ігри. Функції розроблено з використанням адаптивної технології для окремих осіб
* Деякі важливі ігрові показники: DAU/MAU, сеанс, рейтинг завантажень, утримання та показники продуктивності
* Основний ризик тестування мобільних ігор полягає в тому, що воно не створює переконливих вражень для цільової аудиторії

Для звичайного користувача відеогра – це кінцевий додаток, який дає йому певний ігровий процес (геймплей). Але що означає відеогра для розробника?

Для розробника відеогра – це трудомісткий процес повний неточностей, які здатні постійно змінюватися протягом усього процесу розробки. У книзі Джейсона Шрейера «Кров, піт і пікселі» можна виділити кілька пунктів, які призводять до нерівномірності процесу розробки:

1. Інтерактивність ігор підводить розробників до бажання досконало розвивати всі деталі гри, що в свою чергу забирає більше і без того обмежених ресурсів;
2. Технічний прогрес завжди приносить в процес розробки як позитивні аспекти (нові інструментарії, спрощення вже існуючих можливостей), так і негативні (ускладнення процесу оптимізації, дефекти через недоробки або конфлікти нових функцій);
3. Якість гри можливо оцінити тільки погравши в неї, що в свою чергу не дає створити точний план дій.

**Мобільна гра**

У чому ж головна відмінність мобільної гри від звичайної відеогри? Щоб дати вичерпну відповідь на це питання, потрібно декомпозувати гру на наступні частини:

1. Геймплей
2. Керування
3. Графіка
4. Продуктивність/Результативність

Оскільки мобільна платформа обмежена у ресурсах, ідеально потрапити під усі чотири пункти – неможливо. Під час створення складної сучасної графіки у мобільній грі ставиться під загрозу продуктивність, при старанній роботі над детальним геймплеєм можна прийти до проблематики з керуванням. Ігноруючи ці особливості, є ризик створити не дружній до гравця продукт.

В цілому, мобільна гра дає значно менше можливостей для команди, аніж проєкт на ПК, що у свою чергу ускладнює процес розробки.

**Жанри та механіки**

Умовно конфлікт чотирьох пунктів створив певну жанрову та під-жанрову палітри. Можна назвати наступні жанри мобільних ігор:

1. Платформери
2. Шутери
3. Гонки
4. Adventure/Пригоди
5. RPG
6. Аркади
7. Настільні ігри
8. Match 3
9. Hidden Object

Відповідно існують особливі механіки які можуть використовуватися у мобільних іграх:

1. Touch/Swipe/Multi Touch – основні механіки управління на мобільних пристроях, які призводять до схожих простих механік з ПК індустрії (click, drag and drop). Частіше, ці механіки зустрічаються у настільних іграх, в деяких адаптованих RPG, аркадах, Match 3 та Hidden object.
2. Віртуальні геймпади – ця механіка керування дуже відрізняється від механіки звичайного геймпада, з якою можна ознайомитись на консолях або ПК. Оскільки вона створюється з окремих активних елементів, з величезною ймовірністю механіка може конфліктувати з іншими елементами інтерфейсу. Крім того, в більшості реалізацій віртуальні геймпади мають широку кастомізацію, що може тільки збільшити ризики створення дефекту. Дана механіка універсальна та може бути застосована в таких жанрах як RPG, Adventure, Аркади, Гонки, Платформери.
3. Механіки, засновані на гіроскопічних датчиках. Реалізаціями можуть бути як і звичайний автоповорот, який впливає на розташування елементів інтерфейсу, так і повноцінна механіка допомоги прицілювання в деяких шутерах.

**Особливості тестування мобільних ігор**

На відміну від родичів ПК, на мобільних девайсах є певні умовності, які у малому або у великому об'ємі здатні впливати на якість кінцевого продукту.

**Перша особливість мобільних платформ** – розмір екрану.

Зараз існує величезна кількість мобільних девайсів з унікальними розмірами екранів. І якщо умовна текстура моделі в грі на девайсі з маленьким екраном роздільної здатності HD  буде відображена цілком добре, то на великому екрані на умовному планшеті з роздільною здатністю FHD текстура може виглядати посередньо.

Сюди також можна віднести адаптацію інтерфейсу, так як більшість елементів розміщується з точністю по пікселях, в результаті є ймовірність, що на якомусь девайсі зустрінеться накладення об'єктів через іншу роздільну здатність.

А після новаторського кроку від Apple існують ще й унікальні вирізи на екранах, через які певні елементи інтерфейсу можуть стати недоступними і перекритими.

*Перекриття елементів керування вирізом*

**Друга особливість мобільних ігор** – жести, та й сам Touch-інтерфейс у цілому.

Використовуючи відомі механіки Touch/Swipe/Multi Touch, розробники отримують можливість зробити інтуїтивне та зрозуміле керування. Але з великими можливостями приходить велика відповідальність. Кожен елемент управління – це окремий модуль, який може зламатися за допомогою найнесподіваніших способів. Наприклад: кнопка може виявитися не клікабельною, або вічно затиснутою; анімація скрола після свайпа може відображатися ривками; після багаторазового або одночасного натискання декількох різних елементів можна отримати конфлікт функцій.

*Накладання елементів керування*

**Третя особливість мобільних ігор** – порушення у роботі з боку системи.

До цієї категорії можна віднести:

1. Системні жести:
   * виклик навігаційної панелі;
   * виклик статус-бару;
   * виклик панелі повідомлень;
   * фізичні кнопки;
   * рicture in picture;
   * split screen.
2. Системні переривання:
   * вихід з програми;
   * згортання додатку;
   * дзвінки;
   * будильники;
   * увімкнена музика;
   * сповіщення;
   * розрядження батареї.

На жодному із цих пунктів додаток не повинен крешитись. У разі діяльності поза грою повинна включатися пауза, а прогрес гравця повинен відразу ж завантажуватися в хмару або на сервери гри. Після повернення в гру не повинно спостерігатися проблем з продуктивністю, а ігровий прогрес повинен бути таким як і був.

*Гра обрізана у режимі Split Screen*

**Четверта особливість мобільних ігор** – якість мобільного інтернету.

Важливо розуміти, що мобільні пристрої мають два основних типи підключення для виходу в інтернет: WiFi та Mobile Data (Edge, 2G, 3G, 4G і т. д.). Розбираючи кожен тип підключення, можна прийти до висновку, що принцип роботи досить схожий. Фактично, якість і надійність мережі повною мірою залежить від покриття і навантаження, як наслідок – тестувати потрібно як хороші сценарії, так і найгірші. Наприклад, після успішної транзакції користувач може не отримати бажаний контент через те, що робота з сервером була нестабільною, хоча по документації додаток не повинен створювати такі випадки. Або, наприклад, функція програми стає недоступною через обрив зв'язку з сервером при гарній якості мережі.

*Нестабільне підключення*

**П’ята особливість** – продуктивність. Найважливіший, але іноді ігнорований розробниками до останніх стадій розробки, пункт. Сюди можна віднести швидкість запуску ігрових локацій, фреймрейт під час idle сцен, фреймрейт під час завантажених сцен, фреймрейт на складних локаціях.

Процес оптимізації не завжди легкий і вже точно не може бути реалізованим на швидкоруч.

Щоб копнути глибше в складність процесу оптимізації, можна поцікавитися досвідом CD Project Red та їх легендарним портуванням The Witcher 3 під платформу Nintendo Switch. Команда витратила цілий рік на переробку багатьох аспектів, щоб найкраща гра 2015 року без геймплейного вирізання змогла перебратися на мобільну платформу.

Провести тестування продуктивності можна за допомогою спеціально написаних локаційних бенчмарків, або зробити виміри вручну. Для замірів фреймрейта в мобільних іграх використовують вбудований функціонал движка, або ж різного роду FPS monitor додатки, які вимагають детального налаштування. Одиниця виміру фреймрейта – FPS (Frame Per Second, Фрейм/Кадр за секунду).

**Вимоги до тестування мобільних ігор**

Пристрій, на якому буде проводитися тестування, має бути скинуто до заводських налаштувань, щоб повністю переконатися, що воно не конфліктує з попередньо встановленими програмами.

У разі, якщо в мобільній грі є функція взаємодії з іншими додатками, встановлення цих додатків дозволяється, тому що дане тестування відповідає нормам end to end.

Обовязкові до перевірки пункти:

1. Структура білда:
   * у структурі білда присутні іконки під усі актуальні або заявлені у специфікації дозволи екрану;
   * версія SDK, з допомогою якого була зібрана мобільна гра,співпадає із версією SDK максимально підтримуваного девайсу.
2. Під час першого запуску обов'язково перевіряємо:
   * Landing Page – сторінку після запуску програми, де відображаються пропозиції від розробника;
   * Splash Screen – зображення, яке з'являється під час завантаження гри, найчастіше логотип студії розробника або ж видавця;
   * Loading screen – завантажувальний екран, на якому відображаються найчастіше тематичні арти з прогрес баром (іноді розробники додають в цей компонент хінти-підказки по грі).
3. Після завантаження додатку перевіряємо:
   * меню;
   * профіль;
   * налаштування;
   * допомогу;
   * інформацію про видавця.
4. Всередині гри перевіряємо :
   * коректність відтворення звуку;
   * механіки гри;
   * збереження гри.
5. У випадку наявності функції прив’язки соціальних мереж та інших сторонніх сервісів, в обов'язковому порядку необхідно перевірити прив'язку до кожного сервісу.

***Конфігураційне тестування***

Даний вид тестування дозволяє перевірити працездатність програмної системи в умовах різних операційних систем, апаратних та програмних конфігурацій. Ознайомившись із визначенням, можна помітити, що конфігураційне тестування сходиться із визначенням тестування здатності до портування (portability testing), і це неспроста, оскільки дані поняття практично ідентичні.

**Конфігураційне тестування** – ще один із видів традиційного тестування продуктивності. У цьому випадку замість того, щоб тестувати продуктивність системи з погляду навантаження, тестується ефект впливу на продуктивність змін у конфігурації. Хорошим прикладом такого тестування можуть бути експерименти з різними методами балансування навантаження. Конфігураційне тестування також може бути поєднане з тестуванням навантаження, стрес або тестуванням стабільності.

Виходячи із визначення, можна виділити 2 мети конфігураційного тестування:

* Визначити оптимальну конфігурацію устаткування, що забезпечує необхідні характеристики продуктивності та часу реакції системи, що тестується.
* Перевірити об’єкт тестування на сумісність із обладнанням, що заявлене в специфікації, операційними системами та програмними продуктами третіх компаній.

Також, можна виділити 2 рівня проведення тестування конфігурації – клієнтський та серверний. **Клієнтський**. Додаток тестується з позиції робочого оточення кінцевого користувача. А саме:

* Кросплатформенне тестування (типи і версії ОС).
* Кросбраузерне тестування (використовується, при тестуванні веб-додатків).
* Тестування роботи при різних версіях драйверів.
* При тестуванні ігрових додатків – тестування відеоадаптера.

Якщо ж програма **клієнт-серверна**, необхідно протестувати взаємодію додатоку із оточенням:

* Апаратним (тип і кількість процесорів, обсяг пам’яті, характеристики мережі/мережевих адаптерів і т.д.).
* Програмним (ОС, драйвера та бібліотеки, стороннє ПЗ, що впливає на роботу програми і т.д.).

Безпосередньо, саме тестування проводиться таким чином:

* Визначаються всі можливі конфігурації, які необхідно протестувати.
* Дані конфігурації розподіляються у чергу за пріоритетом, так як їх кількість може бути дуже великою.
* Відповідно до встановлених пріоритетів проводиться саме тестування.

Це і є основи конфігураційного тестування. Сподіваюся, Ви збільшили свій арсенал знань новим потужним знанням, яке тепер можна використовувати при працевлаштуванні.