Universal Acceptance

Концепция готовности к универсальному принятию

6 апреля 2020 года

СОДЕРЖАНИЕ

Введение

Универсальное принятие (UA) — это возможность использования идентификаторов, которых не было на заре становления интернета, особенно когда речь идет об интернационализации. Более конкретно и для целей этого документа UA определяется как состояние соответствия приложения или службы требованиям в отношении поддержки следующего:

- Интернационализированные доменные имена (IDN-домены)
    - Потому что некоторые программы ожидают, что доменное имя будет представлено в кодировке ASCII.
- Интернационализация адреса электронной почты (EAI)
    - Потому что некоторые программы ожидают, что локальная часть (например, строка слева от знака «@» для алфавитов с написанием слева направо) адреса электронной почты будет иметь кодировку ASCII.
    - Потому что некоторые почтовые серверы или службы не готовы к получению писем с локальной частью адреса электронной почты в кодировке, отличной от ASCII.
- Длинные строки доменов верхнего уровня (TLD)
    - Потому что некоторые программы ожидают, что длина строки TLD не будет превышать трех символов.
- Добавление и удаление строк TLD
    - Потому что некоторое программное обеспечение проверяет TLD на основе устаревшего и статичного списка возможных TLD.

Что касается добавления и удаления строк TLD, стоит отметить, что вопреки убеждениям некоторых пользователей TLD часто добавляются в корневую зону или удаляются из нее, иногда даже ежедневно. Например, в течение 10-дневного периода в ноябре 2019 года из корня были удалены 7 TLD.

При определении того, правильно ли приложение какого-либо вида на какой-либо платформе поддерживает UA, могут возникнуть трудности. Цель настоящего документа — определить рамки и концепцию для выявления проблем, с которыми сталкивается сообщество разработчиков, чтобы оно могло заниматься их устранением и повышать готовность к UA.

Смежная работа по UA представлена здесь: https://uasg.tech/information/developers/.

Категории приложений

Для целей данного исследования приложения делятся на веб-приложения и собственные, как описано ниже.

Веб-приложение

- Пользовательский интерфейс (UI) находится в браузере.
- Браузер либо автономный, либо встроенный. Обратите внимание, что некоторые приложения кажутся собственными, но на самом деле отображают HTML-страницы с помощью встроенного браузера. Встроенный браузер может находиться в приложениях для настольных ПК, в автомобилях, носимых устройствах, потребительских товарах и т. д.

Собственное приложение

- Компьютерное или на мобильных платформах (сотовый телефон, планшеты и т. д.)
- Работает непосредственно в операционной системе (ОС).
- Преимущественно не использует браузер или встроенный браузер.

Компоненты

Приложение обычно состоит из различных компонентов. У каждой указанной выше категории приложений свой собственный набор компонентов. В этой работе намеренно используется упрощенный подход и основное внимание уделяется только основным компонентам приложения. Это означает, что некоторые приложения модель может описывать не совсем точно. Разделение приложения на компоненты позволяет, занимаясь вопросами соблюдения требований к UA, применять подход на основе различных шлюзов между компонентами, как рассмотрено далее. Такие шлюзы можно рассматривать как функциональное разделение в приложении.

Веб-приложение

Веб-приложение состоит из следующих компонентов:

- Веб-браузер: Браузер — это еще одно приложение; при этом оно служит неотъемлемой составляющей доставки веб-приложения. Обратите внимание, что иногда веб-браузеры встроены в собственное приложение.
- Внешний интерфейс: Внешний интерфейс — компонент, который создает код и обеспечивает выполнение кода в браузере. Он отвечает за создание HTML-страниц и позволяет выполнять в браузере код HTML, CSS и Javascript.
- Серверная часть: Серверная часть — работающий на удаленном сервере компонент, который обычно выполняет более сложные задачи, а также поддерживает работоспособное состояние и обращается к базе данных.
- База данных: База данных — компонент, хранящий информацию на удаленном сервере. База данных может находиться под управлением типичного ядра базы данных SQL или даже иметь вид обычного файла.
- Файловая система: Файловая система — место хранения файлов в операционной системе (ОС).
- Внешняя служба: Иногда приложение использует внешнюю службу для дополнения своих основных функций. Типичным примером внешней службы является служба аутентификации, работающая внутри предприятия или предоставляемая поставщиком облачных услуг.

Аспекты

- Учитывая характер современных веб-приложений, эта модель упрощена, но содержит типичные компоненты, используемые при разработке программного обеспечения, и является достаточной для целей этой работы.
- В современных веб-браузерах предусмотрено локальное хранилище, которое обычно ограничено по размеру и возможностям, но может использоваться разработчиками приложений для хранения идентификаторов UA.
- В современных облачных приложениях внешний интерфейс и серверная часть обычно работают без отслеживания состояния, поэтому их можно динамически развертывать в нескольких экземплярах по горизонтали в зависимости от нагрузки. Следовательно, внешний интерфейс и серверная часть обычно не предусматривают хранение данных приложения.
- Как правило, большинство сайтов разрабатываются как приложения. Поэтому в контексте данной работы сайт считается веб-приложением. Таким образом, предложенную модель можно применить к сайту.

Собственное приложение

Собственное приложение состоит из следующих компонентов:

- Пользовательский интерфейс (UI): Визуальный интерфейс для взаимодействия с пользователем. Этот интерфейс отвечает за ввод пользовательских данных, таких как доменные имена или адреса электронной почты.
- Внутренняя часть: Код, обрабатывающий вводимые пользователем данные и обеспечивающий выполнение функций приложения.
- База данных: База данных — компонент, хранящий информацию приложения. База данных может находиться под управлением типичного ядра базы данных SQL или даже иметь вид обычного файла. База данных может быть локальной или удаленной.
- Файловая система: Файловая система — место хранения файлов в операционной системе (ОС).
- Внешняя служба: Иногда приложение использует внешнюю службу для дополнения своих основных функций. Типичным примером внешней службы является служба аутентификации, работающая внутри предприятия или предоставляемая поставщиком облачных услуг.

Этапы прохождения шлюзов

Сообщество UA определило следующие этапы обработки связанных с UA идентификаторов в приложении:

1. Принятие: как приложение принимает введенные пользователем данные, то есть идентификаторы UA.
2. Валидация: как приложение выполняет валидацию идентификатора UA.
3. Обработка: как приложение обрабатывает идентификатор UA.
4. Хранение: как приложение сохраняет идентификатор UA.
5. Отображение: как приложение отображает идентификатор UA.

В этой работе предлагается добавить этап между сохранением и отображением, чтобы обеспечить полную последовательность шагов, которые можно использовать в качестве шлюзов для проверки соответствия. Новый список имеет следующий вид:

1. Принятие: как приложение принимает введенные пользователем данные, то есть идентификаторы UA.
2. Проверка: как приложение выполняет валидацию идентификатора UA.
3. Обработка: как приложение обрабатывает идентификатор UA.
4. Хранение: как приложение сохраняет идентификатор UA.
5. Обработка: как приложение обрабатывает идентификатор UA после сохранения (обычно для отображения).
6. Отображение: как приложение отображает идентификатор UA.

Следует отметить, что для некоторых приложений часть этапов или шлюзов может быть неактуальной. Например, некоторые приложения могут не хранить идентификаторы, поэтому для них этапы 4 и 5 неуместны.

Моделирование приложений

Моделируются две категории приложений на основе их компонентов и этапов прохождения шлюзов.

В таблицах ниже указано «Да», когда компонент играет некоторую роль на определенном этапе, и «М», когда компонент может играть некоторую роль.

Веб-приложение

	Принятие	Проверка	Обработка	Хранение	Обработка	Отображение
Браузер	Да	Да (только IDN-домены в URL)		M (локальное хранилище)		Да
Внешний интерфейс		M	M		M	
Серверная часть		M	M		M	
База данных				Да		
Файловая система				Да		
Внешняя служба		M	M	M	M	

Собственное приложение

	Принятие	Проверка	Обработка	Хранение	Обработка	Отображение
Пользова-тельский интерфейс	Да	M				Да
Внутренняя часть		M	M		M	
База данных				Да		
Файловая система				Да		
Внешняя служба		M	M	M	M	

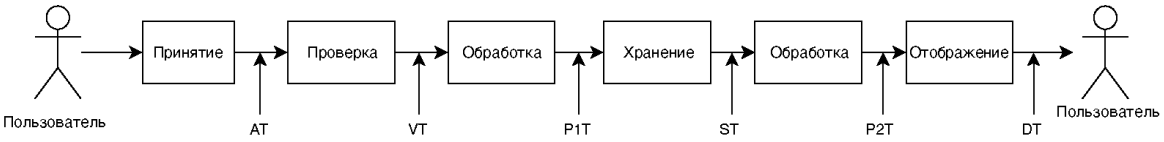
Подход на основе шлюзов

В данной работе для проверки приложения на соответствие требованиям UA предлагается подход на основе шлюзов. Этот подход основан на тестировании различных компонентов на различных этапах, теперь именуемых шлюзами.

Определены следующие категории тестирования.

Ключевое слово	Значение
AT	Тест принятия
VT	Тест валидации
P1T	Тест обработки на входе
ST	Тест хранения
P2T	Тест обработки на выходе
DT	Тест отображения

На иллюстрации ниже показан предлагаемый подход на основе шлюзов. Каждая категория тестирования применяется на разных этапах работы приложения.



Аспекты применения шлюзов

Архитектура современных программ представляет собой сложные системы с контейнерами, микросервисами, многоуровневостью, кэш-памятью на всех уровнях, облачными сервисами и т. д. Фактически, приложение представляет собой сеть компонентов, взаимодействующих друг с другом сложным образом. Из-за такой сложности даже разработчики, участвующие в проекте, испытывают трудности с выявлением проблем.

Хотя этапы (принятие, проверка, обработка, хранение, отображение) определены, некоторые из них можно объединить. Например, проверка и обработка могут происходить одновременно в пределах одного метода объекта идентификатора. В настоящей работе предложена модель, но границы между компонентами зачастую размыты. В архитектуре современных программ часто используются облачные службы разных поставщиков. Код самих служб и их компонентов обычно не раскрывается. Следовательно, эти службы должны рассматриваться как черные ящики без возможности их изучения; они могут быть проверены только на базе интерфейса, который они предоставляют. Учитывая их доминирующее положение на рынке разработки программного обеспечения, их необходимо учитывать при проверке соответствия требованиям UA.

Примерами таких услуг являются сторонние службы идентификации/аутентификации/авторизации. Разработчику программного обеспечения проще использовать Google, Facebook, Apple или другие уже существующие службы аутентификации пользователей, поскольку это позволяет ему сосредоточиться на уникальных аспектах соответствующего сервиса, не тратя время и средства на «изобретение велосипеда» для часто встречающихся функций. Это также повышает удобство работы конечных пользователей. Поскольку домены или адреса электронной почты часто применяются в качестве идентификаторов или резервных идентификаторов, они становятся важной составляющей соответствия

приложения требованиям UA, когда оно использует эти службы. Адреса электронной почты не только используются как уникальные идентификаторы пользователей, но также часто хранятся в виде резервных копий для восстановления учетной записи или сброса пароля, что также должно соответствовать требованиям UA.

Компоненты веб-приложения

В этом разделе перечислены некоторые компоненты веб-приложений с примерами. Этот список не является исчерпывающим.

Категория	**Примеры**	**Тип приложения**
Браузер	Chrome, Firefox, Edge, Safari, Opera, Brave, UC, 360, Sogou, Baidu, Tencent, QQ	Компонент веб-приложения
Веб-сервер	nginx, Apache, lighttpd, tomcat, IIS	Серверный компонент веб-приложения
Сертификаты TLS	openssl, letsencrypt	Серверный компонент веб-приложения
Серверный веб-фреймворк/ библиотеки	Django/Python, Flask/Python, Spring/Java, Express/NodeJS, Ruby on Rails, Lumen/PHP, .Net, Laravel	Компонент веб-приложения
Интерфейсный веб-фреймворк/ библиотеки	Angular, React, Vue.js, Backbone, Ember Bootstrap	Компонент веб-приложения

Приложения

В этом разделе перечислены некоторые приложения (веб- или собственные), которые могут быть кандидатами на тестирование соответствия требованиям UA.

Электронная почта (веб-почта)	Gmail, Xgenplus, Coremail, Hotmail/Outlook/Live, Yahoo, Datamail, Mailrelay	Веб-приложение
Электронная почта (собственный клиент (MUA, MSA))	Gmail, Outlook, Xgenplus, Foxmail	Собственное
Электронная почта (сервер (M*A))	MSexchange, postfix, Dovecot, Xgenplus, courier	Собственное
Электронная почта (поставщик услуг)	Gmail, Hotmail/Outlook/Live, Yahoo, datamail CoreMail, QQ mail; 163/126 mail; China Mobile 139.com	Собственное
Антиспам (AS)	Spamjadoo, Comodo, SpamAssassin, SpamTitan	Собственное
Инструменты ОС	Curl, Wget, nslookup, Dig, Telnet, SSH, Host, Hostname, SCP, Ping, Tracert, PathPing	Собственное

CMS	WordPress, Joomla, Drupal, Typo3, Discuz, Pageadmin	Веб-приложение
Веб-приложения социальных сетей	Facebook, Twitter, Pinterest, Instagram, LinkedIn, WhatsApp, WeChat, Sina Weibo, QQ	Веб-приложение
Приложения социальных сетей	Facebook, Twitter, Pinterest, Instagram, LinkedIn, WhatsApp, WeChat, Sina Weibo, QQ	Собственное
Веб-приложения для электронной коммерции	Amazon, Alibaba, Ebay	Веб-приложение

Стоит отметить, что одно и то же приложение часто поставляется в двух вариантах: веб-версия и собственная версия. Например, сеть Facebook доступна как веб-приложение и как собственное мобильное приложение. Тем не менее, в целях соответствия требованиям UA, они должны рассматриваться как отдельные и разные, поскольку в отношении UA могут вести себя по-разному.

Ожидаемое поведение

В этом разделе описывается ожидаемое поведение каждого шлюза в приложении.

Принятие

- Допустимы все формы: например, IDN-домен в виде U-Label или A-Label.

Проверка

- Нормализовать, если не в кодировке ASCII
- По выбору:
  - Убедиться, что IDN-домен действительный
  - Убедиться, что длина метки не превышает максимальную (> 63 октетов, A-Label в случае IDN-домена)
- Возможно (есть плюсы и минусы):
  - Убедиться, что TLD активен

Обработка на входе

- Не производить усечение
- Не допускать произвольное преобразование в ASCII. Например, преобразование «é» в «e».

Хранение

- EAI: локальная часть — UTF8
- IDN-домен: U-Label или A-Label
- Длинный/новый TLD: как есть; без усечения

Обработка на выходе Отображение

- Следует отображать в UTF8
- Можно также отображать IDN-домен как A-Label

Тесты

В этом разделе перечислены различные высокоуровневые тесты, которые можно применять в каждом из шлюзов. Различные возможные наборы данных описаны в документе UASG-004.

AT: Тест принятия

- EAI: проверьте, принимается ли любой UTF8
- IDN-домен: проверьте, принимается ли любой UTF8
- Длинный/новый TLD: проверьте, принимается ли любой TLD как длинная метка

VT: Тест валидации

- Проверьте, нормализуется ли ненормализованная метка UTF8
- Проверьте, выполняется ли валидация действительного/недействительного доменного имени
- Проверьте, выполняется ли валидация действительного/недействительного IDN-домена
- Проверьте, выполняется ли валидация действительного/недействительного длинного/нового TLD

P1T: Тест обработки на входе

- Убедитесь, что входные данные при обработке не изменяются, кроме случаев нормализации строки.

ST: Тест хранения

- EAI:
    - Убедитесь в правильности соответствующего содержимого базы данных при вводе корректных данных EAI:
      то есть в том, что хранится идентичная либо нормализованная версия локальной части. В случае IDN доменная часть может иметь вид A-Label или U-Label.
    - Убедитесь, что некорректные входные данные EAI не сохраняются в базе данных.
- IDN-домен:
    - Убедитесь, что при вводе корректного IDN-домена в базе данных сохраняется метка A-Label или U-Label.
    - Убедитесь, что некорректный введенный IDN-домен не сохраняется в базе данных.
- Строки длинного/нового TLD:
    - Убедитесь, что введенная строка длинного TLD не усекается в базе данных.
    - Убедитесь, что слишком длинная введенная строка TLD не сохраняется в базе данных.
    - Убедитесь, что при вводе нового TLD в базе данных сохраняется новый TLD.

P2T: Тест обработки на выходе

- При наличии правильного значения в базе данных проверьте правильность значения, которое должно быть отображено в пользовательском интерфейсе: локальная часть в формате UTF8, IDN-домен в формате U-Label, TLD с длинной строкой не усекается, строка нового TLD обрабатывается.

DT: Тест отображения

- При наличии правильного значения в базе данных проверьте правильность значения, отображаемого в пользовательском интерфейсе: локальная часть в формате UTF8, IDN-домен в формате U-Label, TLD с длинной строкой не усекается, новый TLD (начиная с корня) отображается, удаленный TLD (начиная с корня) не отображается.

Нормализация

Нормализация необходима, поскольку одна и та же строка может быть представлена разными кодовыми точками. В Unicode определены различные формы нормализации [UTR15]. В отсутствие конкретных деталей разработчики могут использовать форму нормализации C (NFC).

Предлагаемый подход

Предлагаемый подход к проверки соответствия требованиям UA состоит в следующем:

- Определите компоненты, используемые в приложении.
- Проверьте соответствие каждого компонента требованиям UA.
- Протестируйте приложение на соответствие требованиям UA с точки зрения ввода/вывода, не заглядывая «под капот».
- Если результаты тестирования отрицательные, изучите ситуацию, применив подход на основе шлюзов к интерфейсам различных компонентов, как описано ранее.

Следует отметить, что компонент может не соответствовать требованиям UA, но это не означает, что все приложение не соответствует требованиям UA. Это связано с тем, что разработчики приложения могли добавить еще один слой над базовым компонентом для поддержки UA. Например, как показало тестирование библиотек Java на соответствие требованиям UA, базовая JRE JAVA не совместима с UA, поскольку поддерживает IDNA2003. Однако разработчик может использовать и добавить новую библиотеку, которая поддерживает UA, поэтому тот факт, что приложение реализовано на Java, не означает, что оно не соответствует UA.

Эта концепция предназначена для тестирования программных приложений. В том виде, как есть, она может быть неприменима в ряде других сценариев использования, таких как библиотеки или компоненты инфраструктуры. Обмен SMS-сообщениями — пример инфраструктуры, которая обеспечивает возможность передачи интернационализированных идентификаторов, но не обязательно укладывается в рамки данной концепции.

Веб-приложение: Подкатегория CMS

Ниже показано применение подхода к категории веб-приложений, являющихся системами управления контентом (CMS). Это неполный список CMS, но в нем отражены наиболее популярные системы с ранжированием по различным сайтам. Лидером рынка является WordPress.

Все они с открытым исходным кодом. Этот код можно загрузить и установить на сервере, а во многих случаях также использовать в службах хостинга.

	Браузер	Внешний интерфейс	Серверная часть	База данных	Шлюзы/контрольные точки тестирования
WordPress	Любой	PHP	PHP/Apache	MySQL/MariaDB	AT, VT, P1T, ST, P2T, DT
Joomla	Любой	PHP	PHP/Apache	MySQL/MariaDB	AT, VT, P1T, ST, P2T, DT
Drupal	Любой	PHP	PHP/Apache /nginx	MySQL/MariaDB/ PostgreSQL/SQLite	AT, VT, P1T, ST, P2T, DT
Typo3	Любой	PHP	PHP/Apache /nginx	MySQL/MariaDB/ PostgreSQL/SQLite	AT, VT, P1T, ST, P2T, DT

Видно, что они имеют одинаковую архитектуру и одинаковые компоненты. Ко всем можно применить одинаковые тесты.

Резюме

В этой работе предложена концепция для определения объема работ по оценке соответствия приложений требованиям UA. Определены две категории приложений и серия шлюзов для проверки приложения с помощью различных тестов.

Ссылки

[UASG] https://uasg.tech
[UASG-004] Сценарии использования для оценки готовности к универсальному принятию, апрель 2017 года, https://uasg.tech/wp- content/uploads/documents/UASG004-en-digital.pdf
[UTR15] Формы нормализации Unicode, стандарт Unicode®, приложение № 15, https://unicode.org/reports/tr15/