МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА ПО ДИСЦИПЛИНЕ

«ПРОЕКТНАЯ ДЕЯТЕЛЬНОСТЬ»

Лаборатория Тестирования

Куратор проекта: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / Яковлев Станислав Игоревич, Преподаватель/

подпись ФИО, уч. звание и степень

Студент: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / Мироненко Вадим Викторович, 201-321/ подпись ФИО, группа

Студент: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / Енилеев Ислам Русланович, 201-321/ подпись ФИО, группа

Студент: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / Адигамов Руслан Равшанович, 201-321/

подпись ФИО, группа

Студент: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / Белов Павел Денисович, 181-331/

подпись ФИО, группа

Студент: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / Кузина Валерия Михайловна, 191-362/

подпись ФИО, группа

Москва, 2021

ВВЕДЕНИЕ

Цель профессии QA-инженера — помочь создать качественный продукт путём комплексного тестирования и выявления недостатков. Основная задача QA-инженера — локализовать дефекты и, следовательно, обеспечить высокое качество процесса разработки и его результатов.

Лаборатория тестирования позволяет студентам изучить базовые основы деятельности QA-инженера с помощью тривиальных заданий, на основе которых формируются тестовые технические задания для устройства на работу в той или иной IT-компании. Освоение основ, подготовленных на данном проекте, позволяет получить быстрый старт в изучении обеспечения качества (Quality Assurance). Выполненные за семестр задания представлены ниже.

1 ТЕСТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ - процесс исследования, испытания программного продукта, имеющий своей целью проверку соответствия между реальным поведением программы и её ожидаемым поведением на конечном наборе тестов.

Виды тестирования ПО:

* Функциональное тестирование
* Системное тестирование
* Тестирование производительности
* Регрессионное тестирование
* Модульное тестирование
* Тестирование безопасности

1.1 Тестовая документация - документ или совокупность документов, расписывающих все проведенные тесты в пределах одного проекта; все работы, проводимые командой тестирования.

Тестовая документация включает в себя: тест план; тестовая стратегия; чек-лист; тестовый сценарий; тестовый комплект; пользовательская история; отчет о дефекте.

1.2 Описание дефектов - краткое описание бага, которое включает в себя информацию о стенде, шагах воспроизведения, фактическом и ожидаемом результате.

2 ТЕСТИРОВАНИЕ ВЕБ-СЕРВИСОВ - это тип тестирования программного обеспечения, который проверяет веб-сервисы. Целью тестирования веб-служб является проверка функциональности, надежности, производительности и безопасности API (интерфейса прикладных программ). Основные виды тестирования веб-сервисов:

* Функциональное тестирование.
* Тестирование совместимости.
* Тестирование производительности.
* Тестирование безопасности.
* Тестирование локализации.
* Тестирование соответствия.

2.1 Тестирование API - это тип тестирования программного обеспечения, который включает тестирование интерфейсов прикладного программирования, чтобы определить, соответствуют ли они ожиданиям функциональности, надежности, производительности и безопасности.

К тестированию API применимы следующие типы тестов:

* Функциональное тестирование – тесты должны выполнить набор вызовов, задекларированных в API, чтобы проверить общую работоспособность системы.
* Usability-тестирование – проверяет, является ли API функциональным и обладает ли удобным интерфейсом, также проверяется интеграция с другими.
* Тестирование безопасности – проверяет используемый тип аутентификации и шифрование данных с помощью HTTP.
* Автоматизированное тестирование – создание скриптов, программ или настройка приложений, которые смогут тестировать API на регулярной основе.
* Тестирование документации – проверяется полнота описаний функций API, её понятность

2.2 Тестирование WEB - тип тестирования ПО, ориентированный на WEB-продукты (для взаимодействия с которыми необходим доступ в сеть)

Виды тестирования WEB:

* Функциональное тестирование
* Тестирования удобства пользования (юзабилити)
* Тестирование производительности
* Тестирование интерфейса пользователя (UI testing)
* Тестирование безопасности.

2.3 Нагрузочное тестирование - тестирование производительности, сбор показателей и определение производительности и времени отклика программно-технической системы или устройства в ответ на внешний запрос с целью установления соответствия требованиям, предъявляемым к данной системе.

Задача тестирования производительности - определение масштабируемости приложения под нагрузкой, при этом происходит:

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

2.4 Автоматизированное тестирование - это процесс проверки программного обеспечения, который включает проведение таких основных функций и шагов теста, как запуск, инициализация, выполнение, анализ и выдача результата, автоматически посредством специализированных инструментов.

3 ТЕСТИРОВАНИЕ МОБИЛЬНЫХ ПРИЛОЖЕНИЙ – это процесс, с помощью которого прикладное ПО, разработанное для портативных мобильных устройств, проверяется на его функциональность, удобство использования и совместимость. Существует несколько основных типов тестирования мобильных приложений:

* Нагрузочное тестирование или тест на производительность.
* Тестирование на восстановление.
* Интеграционное тестирование.
* Тестирование функциональности.
* Тестирование безопасности.
* Юзабилити–тестирование.
* Тестирование совместимости.

3.1 Ручное тестирование - это человеческий подход, анализ или оценка. Этот подход ориентирован на пользователя и фокусируется на исследовательских способах мониторинга того, отвечает ли мобильное приложение требованиям и ожиданиям пользователя, проверить на предмет внешнего вида и удобства использования.

3.2 Автоматизированное тестирование -  вместо того чтобы вручную проводить рутинные трудоёмкие процедуры, можно делегировать значительную их часть фреймворкам. Автоматизация упрощает проверку и помогает ускорить регрессионное тестирование, а также даёт возможность использовать ранее недоступные типы тестирования.

Перечень инструментов для автоматизации:

**Android:**

* Espresso
* UI Automator

**iOS:**

* XCUITest
* EarlGrey

**Универсальные:**

* Detox
* Appium
* Ranorex
* TestComplete Mobile

4 ЮЗАБИЛИТИ ТЕСТИРОВАНИЕ - исследование, выполняемое с целью определения, удобен ли некоторый объект (например, определенный интерфейс) для его предполагаемого применения.

Нужно проверить приложение по правилам «правильного» пользовательского интерфейса:

1) **Информативность системы** – пользователь всегда должен знать текущий статус приложения.

2) **Приближенность приложения к реальному миру** – диалог с пользователем должен вестись на понятному ему языку, остерегаясь использования непонятной терминологии.

3) **Система должна иметь выходы** – приложение всегда должно иметь «запасные выходы» из любой функциональности, которые пользователь по ошибке запустил.

4) **Однозначность**. Все термины, функции и понятия должны описываться в едином толковании – у пользователя не должно возникнуть путаницы.

5) **Предусмотрительность**. Система должна всячески «оберегать» пользователя от возможных ошибок.

6) **Наглядность**. Пользователь не должен ломать голову в попытках понять, что ему нужно делать или пытаясь вспомнить, как он достиг того или иного состояния системы. Возможные манипуляции с программой должны быть постоянно наглядными.

7) **Гибкость и эффективность**. Предоставляйте опытным пользователям возможность избегать рутинных действий, и в то же самое время, необходимо скрывать расширение функционала от неопытных.

8) **Лаконичность и точность**. Диалоги должны содержать только ту информацию, которую необходимо донести до пользователя, ничего лишнего.

9) **Лояльность к ошибкам**. Информация об ошибках должна быть понятной и содержать подсказки к дальнейшим действиям.

10) **Постоянная справка**. Как бы информативно не была спроектирована система – она всегда должна содержать раздел справки и документации.

5 Менеджмент тестирования - это решение для управления качеством, предназначенное для совместной работы. Оно предоставляет возможности детального планирования тестирования и управления ресурсами тестирования на всех этапах — от определения требований до выявления дефектов.

5.1 Управление тестированием - существуют системы хранения информации о том, как должным образом проводить тестирование, осуществление очередности проведения тестирования в соответствии с его планом, а также для получения информации в виде отчетов о стадии тестирования и качестве тестируемого продукта.

Самые популярные системы управления тестированием:

* ALM Octane
* Test IT
* TestRail
* Zephyr
* Allure EE
* TM4J
* Qase
* PractiTest
* Testuff
* Azure
* MTM TFS
* Kualitee

5.1.1 Работа в Jira - Jira Software является продуктом, разработанным с целью упростить управление рабочим процессом для самых разных команд. Изначально система Jira создавалась как решение для отслеживания задач и ошибок. Но сегодня Jira — это мощный инструмент управления работой, подходящий для самых разных случаев, от управления требованиями и сценариями тестирования до agile-разработки программного обеспечения.

Вот для чего используется Jira:

* для управления требованиями и сценариями тестирования
* для agile-команд
* для команд управления проектами
* для команд разработки ПО
* для команд управления продуктами
* для управления задачами
* для отслеживания ошибок

5.1.2 Работа в Trello - Trello является универсальным инструментом для ведения рабочих и личных проектов. Он позволяет отслеживать выполнение каждой задачи, координировать работу нескольких человек, следить за сроками и хранить всю необходимую информацию в одном месте.

**Списки (или колонки) Trello**

Доска Trello не имеет смысла, если в ней нет списков-колонок. Именно по ним будут путешествовать задачи в ходе выполнения.

Обычно в качестве колонок выступают:

* **Статусы задач**. Например, «Сделать», «В работе», «В доработке», «Сделано».
* **Время**. Например, «Вчера», «Сегодня», «Завтра», «Когда-нибудь потом» или «Январь», «Февраль», «III квартал 2020».
* **Приоритет**. Например, «Срочно», «Выполнить в течение 2 дней», «Выполнить в течение недели».
* **Виды задач**. Например, «SEO», «Дизайн», «Контент», «Разработка».
* **Этапы**. Например, «Первый звонок», «Второй звонок», «Встреча», «Сделка», «Лиды».

**Информация в карточках Trello**

1. **Название**. Например, «Добавить всплывающее окно на главную».

2. **Описание**. Например, «Всплывающее окно с предложением подписаться на почтовую рассылку. Активируется при скролле (75%)».

3. **Участник или исполнитель**. Например, «Петя Птичкин». Их может быть несколько человек.

4. **Срок**. Например, 23 февраля 21:00. При добавлении срока в карточке появляется чек-бокс о выполнении.

5. **Чек-лист**. Например, «1. Получить текст от копирайтера. 2. Создать попап. 3. Написать маркетологу».

6. **Обложка**. Если добавить к карточке изображение, оно автоматически станет обложкой и выделит карточку в колонке. Это можно отменить в настройках карточки.

7. **Метка**. Метки — это цветовые ярлыки для задач. Ярлыками можно разделять задачи по их типу, сложности, важности, отделу или вашей личной симпатии к задаче.

8. **Комментарии**. Комментарий к карточке может оставить любой участник доски. Например, «Не успею выполнить к 21:00».

5.1.3 Работа в Redmine - Redmine считается открытым серверным веб-приложением для управления проектами и задачами (в том числе для отслеживания ошибок), которое включает в себя диаграммы Ганта, календарь, вики, форумы, настройку ролей и уведомления по электронной почте.

Возможности Redmine:

* Учет рабочего времени.
* Наличие диаграммы Ганта.
* Поддержка различных СУБД.
* Разные схемы аутентификации.
* Возможность создания своих полей.
* Поддержка языков, в частности, русского.
* Отдельные форумы для каждого проекта.
* Возможность работы с документами и файлами.
* Работа на несколькими проектами одновременно.
* Распределение ролей и обязанностей. Настройка прав доступа.

5.2 Бизнес-ценность тестирования:

1. **Экономия времени и средств** **(для автоматизированного)**: написание автоматических тестов может занимать больше времени, чем письменные тесты. Однако, учитывая, что тесты выполняются несколько раз, предельная работа для выполнения автоматических тестов на несколько порядков меньше. То, что автоматизированные тесты дешевы, облегчает изменение системы с течением времени.
2. **Документация**: нет лучшего способа узнать, как работает система, чем ее тесты. Любая другая документация обычно устарела с момента ее написания, но тесты (по крайней мере, те, которые прошли) показывают, как все работает на самом деле. Это верно как для конечного пользователя, так и для документации API.
3. **Качество кода**: тестовая запись заставляет вас:

* считать клиентов, потому что тесты являются клиентом
* ломает зависимости, где тестирование кода часто означает выяснение того, как сделать этот код не требующим доступа к какой-либо другой большой системе

5.3 Управление дефектами - это систематический процесс выявления и устранения ошибок. Цикл управления дефектами содержит следующие этапы:

1. Обнаружение дефекта
2. Категоризация дефекта
3. Устранение дефекта разработчиками
4. Проверка тестерами
5. Закрытие дефекта
6. Отчеты о дефектах в конце проекта

6 ОТЧЕТ ПО ТЕСТИРОВАНИЮ ПРОЕКТА

<https://docs.google.com/document/d/1LR7ZCyLOuFi88yJEE-jcEYikTxzl7nMBLPT_7u0cM1U/edit> - ДЗ-1

<https://docs.google.com/document/d/1phy54J9ZcgAKiCm7DiED51hbePnv2htKr_v8lGxqOfU/edit?usp=sharing> - ДЗ-2

<https://docs.google.com/document/d/1MhydCpa028iE-9IV_l9bWqkAvpa6ZcVrmlDfAVyAvd8/edit?usp=sharing> - ДЗ-3

<https://docs.google.com/document/d/1iBCe7_t_dHQXX_twVeDorqgnCPEIDwWdbVR3grX_ZF0/edit?usp=sharing> - ДЗ-4

<https://drive.google.com/drive/folders/1our0I7pNxlDHzgT7HSUC1ngrQMJ0yop2?usp=sharing> - ДЗ-5

<https://github.com/6az00ka/limeteam4spring> - git-репозиторий

ЗАКЛЮЧЕНИЕ

Список источников:

<https://qalight.ua/ru/baza-znaniy/chto-takoe-testirovanie-programmnogo-obespecheniya/>

<http://www.protesting.ru/testing/>

<https://logrocon.ru/news/defects>

<http://www.protesting.ru/testing/types/loadtesttypes.html>

<https://www.performance-lab.ru/blog/load-testing/testirovanie-proizvoditelnosti>

<https://www.dmosk.ru/terminus.php?object=redmine>

<https://www.atlassian.com/ru/software/jira/guides/use-cases/what-is-jira-used-for>

<https://www.houzz.ru/statyi/produktivnosty-trello-plaginy-dlya-domashnih-del-stsetivw-vs~98438061>

<https://www.edsd.ru/ru/uslugi/testirovanie_po>

<https://qastack.ru/software/90015/automated-testing-explaining-its-business-value>

<https://habr.com/ru/hub/web_testing/>

<https://www.a1qa.ru/blog/vidy-testirovaniya-web-prilozhenij-kak-vybrat/>

<https://www.software-testing.ru/library/testing/testing-automation/3382-api-testing>

<http://33testers.blogspot.com/2015/07/api.html>

<https://habr.com/ru/post/522474/>

<https://quality-lab.ru/blog/the-purpose-of-test-documentation/>

<https://habr.com/ru/company/touchinstinct/blog/334660/>

<https://zhenyabrik.gitbooks.io/matrixcopy/content/lektsiya-4-ch1-testovaya-dokumentatsiya.html>

<https://qalight.ua/ru/baza-znaniy/yuzabiliti-2/>

<https://topuser.pro/uspeshnoe-yuzabiliti-testirovanie-saita-metodi/>

<https://www.learnqa.ru/manual>

<https://habr.com/ru/company/simbirsoft/blog/459292/>

<https://logrocon.ru/news/defects>