МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

(ФГБОУ ВПО МПУ)



Пояснительная записка по дисциплине

«Проектная деятельность»

По теме: «Лаборатория тестирования. Тестирование приложения “Юла”. Модуль: сообщения и чаты»

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

подпись ФИО, уч. звание и степень

Студент: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / Кирса Максим, 171-371

Студент: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / Варламов Андрей, 171-371

Студент: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / Виноградов Николай, 171-371

Студент: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / Кузнецов Александр, 171-372

Студент: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / Фурман Кирилл, 181-331

Студент: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / Гетун Дмитрий, 181-352

Москва, 2021

# СОДЕРЖАНИЕ

[**ВВЕДЕНИЕ** 3](#_Toc61743304)

[**ПЛАН РАБОТ** 6](#_Toc61743305)

[**1 ТЕСТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ** 8](#_Toc61743306)

[1.1 Тестовая документация 8](#_Toc61743307)

[1.2 Описание дефектов 9](#_Toc61743308)

[**2 ТЕСТИРОВАНИЕ ВЕБ-СЕРВИСОВ** 11](#_Toc61743309)

[2.1 Тестирование API 11](#_Toc61743310)

[2.2 Тестирование WEB 13](#_Toc61743311)

[2.3 Нагрузочное тестирование 15](#_Toc61743312)

[2.4 Автоматизированное тестирование 17](#_Toc61743313)

[**3 ТЕСТИРОВАНИЕ МОБИЛЬНЫХ ПРИЛОЖЕНИЙ** 19](#_Toc61743314)

[3.1 Ручное тестирование 20](#_Toc61743315)

[3.2 Автоматизированное тестирование 22](#_Toc61743316)

[**4 UX ТЕСТИРОВАНИЕ** 24](#_Toc61743317)

[**5 МЕНЕДЖМЕНТ ТЕСТИРОВАНИЯ** 28](#_Toc61743318)

[5.1 Управление тестированием 28](#_Toc61743319)

[5.2 Бизнес-ценность тестирования 28](#_Toc61743320)

[5.3 Управление дефектами 30](#_Toc61743321)

[**6 ОТЧЕТ ПО ТЕСТИРОВАНИЮ ПРОЕКТА** 31](#_Toc61743322)

[6.1 Описание проекта 31](#_Toc61743323)

[6.2 Чек-лист 31](#_Toc61743324)

[6.3 Тест-кейсы 32](#_Toc61743325)

[6.4 Найденные дефекты 37](#_Toc61743326)

[**ЗАКЛЮЧЕНИЕ** 38](#_Toc61743327)

[**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ** 39](#_Toc61743328)

# ВВЕДЕНИЕ

Проект по тестированию мобильного приложения «Юла» от разработчиков «Майл.ру» представляет собой поиск ошибок в работе этого самого приложения, которые могут помешать пользователю комфортно использовать данный программный продукт.

В данном случае нашей командой проанализирована работа средств коммуникации, а именно чатов и сообщений в приложении.

Данный курс предназначен для ознакомления с таким понятием как тестирование.

Цель курса: ознакомиться с базовыми понятиями и методиками, применяемыми в тестировании, изучить тестируемое приложение, написать чек-листы, тест-кейсы и рассмотреть выявленные ошибки.

Для учета обнаруженных ошибок будут использоваться такие программные средства как:

* TestReil – инструмент для управления основными процессами тестирования, начиная от регистрации и заканчивая сроками решения и выполнения работы.
* Google Таблицы – для ведения учёта основных аспектов работы.
* Trello – программа для менеджмента и учёта сроков выполнения основных работ.

Данное исследование очень актуально, так как тестирование приложения напрямую влияет на выявление ошибок и критических сбоев в работе приложения, а также помогает обезопасить приложение, сохранить деньги компании и представить пользователю хороший качественный продукт.

## Аннотация

В данной работе представлены методологии и методы исследования предметной области тестирования мобильных приложений. Также представлены тест-кейсы и чек-листы протестированного мобильного приложения раздела чаты и сообщения, с представлением найденных дефектов.

Ссылка на промо ролик: <https://youtu.be/2fgMkY86MyE>

Ссылка на видеопрезентацию: <https://youtu.be/18XHC2b6JFE>

## Участники и их роли

Кирса Максим №группы 171-371 - тимлид, тестировщик iOS, Фурман Кирилл №группы 181-331 - разработка сайта, тестировщик iOS, Кузнецов Aлександр №группы 171-372 - разработка постера, тестировщик iOS, Варламов Андрей №группы 171-371 - тестировщик Android, разработка сайта, Виноградов Николай №группы 171-371 - тестировщик Android, разработка сайта, Гетун Дмитрий №группы 181-352 - тестировщик Android.

Общий план работ представлен на таблице 1 «Общий план работ».

Индивидуальные планы участников:

Кирса Максим №группы 171-371 – создание отчета (6ч), тестирование iOS версии приложения (1ч), подготовка презентации (2ч), запись видеопрезентации (1ч), редактирование Git (10мин). Итого: 76ч

Фурман Кирилл №группы 181-331 – создание отчета (6ч), тестирование iOS версии приложения (1ч), подготовка презентации (2ч), создание сайта(14ч), редактирование Git (10мин). Итого: 90ч

Кузнецов Aлександр №группы 171-372 – создание отчета (6ч), тестирование iOS версии приложения (1ч), создание постера (10ч), редактирование Git (10мин). Итого: 84ч

Варламов Андрей №группы 171-371 – создание отчета (6ч), тестирование Android версии приложения (1ч), создание сайта (12ч), редактирование Git (10мин). Итого: 86ч

Виноградов Николай №группы 171-371 – создание отчета (6ч), тестирование Android версии приложения (1ч), создание сайта (12ч), редактирование Git (10мин). Итого: 86ч

Гетун Дмитрий №группы 181-352 – создание отчета (6ч), тестирование Android версии приложения (1ч), редактирование Git (10мин). Итого: 75ч

# ПЛАН РАБОТ

|  |  |  |  |
| --- | --- | --- | --- |
| **Тема занятия** | **Кол-во часов** | **Из них практических** | **Кол-во часов самостоятельной работы** |
| Введение в тестирование |  |  |  |
| Основные понятия | 18 | - | - |
| Знакомство с Jira, Redmine, Trello. | 12 | 12 | - |
| Тестирование в жизненном цикле разработки ПО. | 12 | 12 | - |
| Тестовая документация. Test Rail. | 12 | 12 | - |
| Составление чек-листов и тест-кейсов. | 12 | 12 | - |
| Правила составления отчетов об ошибках. | 12 | 12 | - |
| Техники тест-дизайна. | 12 | 12 | - |
| Клиент-серверная архитектура. Тестирование API |  |  |  |
| Краткий обзор сети | 12 | - | - |
| Клиент-серверная архитектура  Протокол HTTP | 12 | 12 | - |
| Микросервисная архитектура | 12 | 12 | - |
| Тестирование API с помощью Postman | 12 | 12 | - |
| Тестирование WEB-приложений |  |  |  |
| Основы HTML, CSS, JS для тестировщиков | 12 | 12 |  |
| Особенности тестирования WEB-приложений | - | 12 | 12 |
| DEV-Tools Google Chrome | 6 | 12 | 6 |
| Тестирование мобильных приложений |  |  |  |
| Особенности тестирования мобильных приложений. | 12 | 12 | - |
| Тестирование Android приложений | 6 | 12 | 18 |
| Среда разработки Android Studio и ADB | 6 | 12 | 6 |
| Тестирование мобильных приложений iOS | 6 | 6 | 18 |
| Сбор и анализ трафика (Charles Proxy, Fiddler) | 6 | 18 | - |
|  |  |  |  |
| Подготовка отчета |  |  |  |
| Создание отчета |  |  | 36 |
| Создание сайта | - | - | 38 |
| Подготовка постера | - | - | 10 |
| Подготовка презентации | - | - | 4 |
| Запись видеопрезентации |  | - | 1 |
| Подготовка Git | - | - | 1 |

Таблица 1 «Общий план работ»

# 1 ТЕСТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Тестирование – процесс, содержащий в себе все активности жизненного цикла, как динамические, так и статические, касающиеся планирования, подготовки и оценки программного продукта и связанных с этим результатов работ, с целью определить, что они соответствуют описанным требованиям, показать, что они подходят для заявленных целей и определения дефектов.

Тестирование необходимо, чтобы понять, насколько продукт соответствует предъявляемым к нему требованиям, вписывается ли в общую концепцию видения. Тестирование рассматривается как один из аспектов обеспечения качества.

Тестировщик – специалист, который занимается тестированием. В его обязанности входит поиск возможных ошибок и сбоев в функционале тестирования объекта, например приложения. Тестировщик моделирует ситуации, вероятные при использовании тестируемого объекта, чтобы потом разработчики могли устранить обнаруженные неполадки.

Баг (Дефект) – недостаток в компоненте или системе, способный привести к ситуации сбоя или отказа.

Необходимо понимать разницу между дефектом и ошибкой. Ошибка возникает вследствие действий пользователя, которые не предусмотрены в логике работы системы. Дефект же – это недостаток в самой системе, когда при корректных действиях пользователя возникает некорректный результат.

## 1.1 Тестовая документация

Среди видов тестовой документации выделяют:

* Требования. Регламент того, какие функции должно выполнять приложение в процессе решения полезной для пользователя задачи. Требования делятся на бизнес-требования, пользовательские и системные. Бизнес-требования описывают основные цели разработки, а также бизнес-задачи, которые должна решать система. Пользовательские требования – функции, которые должна иметь система с точки зрения пользователя. Системные требования – описание основных условий, необходимых для запуска и работы ПО. Например, аппаратное обеспечение;
* Тест-план. Документ, в котором описывается объем работ, используемые подходы к тестированию, ресурсы, расписание, а также определяются функции и модули для тестирования, тестовые задачи, ответственные лица и необходимые метрики. Необходим для эффективного распределения работ, понимания итогового объема работы и временных рамок;
* Чек-листы и тест-кейсы. Чек-лист – это набор (список) проверок. Создается и используется, как напоминание о важных тестах. Тест-кейс – полноценный тестовый сценарий для определенной функции, содержащий набор предварительных условий, входных данных, действий (где применимо), ожидаемых результатов и постусловий, разработанных на тестовых условиях.

## 1.2 Описание дефектов

Отчет о дефекте - документ, описывающий обнаруженный дефект, а также содействующий его устранению.

Отчеты о дефектах, как правило, заводятся в специальных баг-трекинговых системах. В отчет вносятся следующие данные:

* Описание ошибки
* Шаги для воспроизведения;
* Фактический результат;
* Ожидаемый результат;
* Серьезность дефекта;
* Приоритет дефекта;
* Статус.

Серьезность дефекта (severity) показывает степень ущерба, который наносится проекту существованием дефекта. Выделяют следующие градации серьезности:

* Block (блокирующий дефект) –блокирует большую часть работы в проекте;
* Crash (авария) –приводит к падению приложения (или даже операционной системы);
* Major (значительный дефект) –дефект в важном, но не ключевом функционале;
* Minor (незначительный дефект) –дефект в дополнительном функционале;
* Tweak (неудобство) –неудобство в использовании,требуется небольшая доработка для повышения удобства использования;
* Text (текст/опечатка) –небольшая текстовая ошибка, опечатка;
* Trivial (тривиальный дефект) –незначительная недоработка;
* Приоритет дефекта, срочность (priority). Показывает, как быстро дефект должен быть устранен. Срочность может быть высокой (High), средней (Medium), низкой (Low).

# 2 ТЕСТИРОВАНИЕ ВЕБ-СЕРВИСОВ

## 2.1 Тестирование API

Каждый раз, когда пользователь посещает какую-либо страницу в сети, он взаимодействует с API удалённого сервера. API — это составляющая часть сервера, которая получает запросы и отправляет ответы.

HTTP – протокол для обмена текстовыми сообщениями, описывает взаимодействие между двумя компьютерами (клиентом и сервером), построенное на базе сообщений, называемых запрос (Request) и ответ (Response). Каждое сообщение состоит из трех частей: стартовая строка, заголовки и тело. При этом обязательной является только стартовая строка.

HTTPS – версия HTTP, использующая шифрование данных.

Толстый клиент для выполнения операций использует вычислительные мощности машины, на которой он установлен. Пример: почтовые программы, например Microsoft Outlook, Mozilla Thunderbird.

Тонкий клиент использует мощности удаленного компьютера, предоставляя пользователю только интерфейс для взаимодействия с ним. Пример: электронная почта в браузере.

Так как RESTful-сервис основан на HTTP, он использует те же стандартные методы протокола:

* GET. Используется для получения содержимого указанного ресурса. В тело ответа может содержаться дополнительная информация;
* HEAD. Аналогичен методу GET, за исключением того, что в ответе сервера отсутствует тело;
* POST. Применяется для передачи пользовательских данных заданному ресурсу;
* PUT. Обновляет содержимое указанного ресурса;
* PATCH. Аналогично PUT, но обновляет данные выборочно, а не целиком;
* DELETE. Удаление содержимого указанного ресурса;
* TRACE. Возвращает полученный запрос так, что клиент может увидеть, какую информацию промежуточные серверы добавляют или изменяют в запросе.;
* OPTIONS. Преобразует соединение запроса в прозрачный TCP/IP-туннель, обычно чтобы содействовать установлению защищенного SSL-соединения через нешифрованный прокси;
* CONNECT. Используется для определения возможностей веб-сервера или параметров соединения для конкретного ресурса.

Когда сервер возвращает HTTP ответ, в нем обязательно содержится код состояний. Примеры некоторых из них.

* 200 ОК – ответ успешно получен;
* 201 Created – элемент создан;
* 204 No Content – нет содержимого. Может быть прислан сервером, если в результате POST или DELETE-запроса по той или иной причине не был обработан ресурс, но URI корректен, запрос корректен;
* 400 Bad Request – задан неверный запрос;
* 401 Unauthorized – пользователь не авторизован;
* 403 Forbidden – доступ запрещен Если данному пользователю запрещен доступ к запрашиваемому ресурсу, то вернётся код 403. Он означает, что аутентификация прошла успешно, но у пользователя нет соответствующих прав;
* 404 Not Found – не найдено, ресурс по данному URI отсутствует;
* 404. Во-первых, этот код возвращается, когда задан действительно неверный адрес и по нему в принципе нет (и не должно быть) ресурса;
* 405 Method Not Allowed – метод не разрешен Данный код ответа может прийти в том случае, если на URI ресурса отправляется запрос с методом, который там применяться не может;
* 409 Conflict – конфликт. Этот код ответа может отправляться в ответ на POST/PUT/PATCH-запросы, если невозможно создать или обновить указанную запись. Например, если какое-то поле должно быть уникальным.

Simple Object Access Protocol – основанный на XML протокол, предназначенный для обмена информацией в распределенных системах. SOAP устанавливает стандарт взаимодействия клиент-сервер и регламентирует,как должен осуществляться вызов, как должны передаваться параметры и возвращаемые значения.

REST (сокр. от англ. Representational State Transfer – «передача состояния представления») – архитектурный стиль взаимодействия компонентов распределенного приложения в сети. REST представляет собой согласованный набор ограничений, учитываемых при проектировании распределенной системы. REST не является стандартом сам по себе, это архитектурный стиль.

RESTful-сервисы работают с объектами на сервере, которые называются «ресурсы». Эти сервисы позволяют внешним системам получать доступ к текстовым представлениям ресурсов и управлять ими с помощью определенного набора операций. Зачастую это стандартные действия: «создать», «прочитать», «обновить», «удалить».

Для тестирования API необходимы инструменты, где можно отправить входные данные в запросе и проверить точность выходных данных. Одним из таких инструментов является Postman. Его возможности:

* Написание и отправка запросов;
* Сохранение запросов;
* Добавление контрольных точек к вызову API;
* Создание разных окружений для одних и тех же запросов;
* Запуск коллекций и использование их как автоматизированных тестов.

## 2.2 Тестирование WEB

Веб-приложение – это клиент-серверное приложение, в котором клиентом выступает браузер, а сервером – веб-сервер (в широком смысле). Основная часть приложения, как правило, находится на стороне веб-сервера, который обрабатывает полученные запросы в соответствии с бизнес-логикой продукта и формирует ответ, отправляемый пользователю. На этом этапе в работу включается браузер, именно он преобразовывает полученный ответ от сервера в графический интерфейс, понятный рядовому пользователю.

Основные виды тестирования сайта (веб-приложения)

1. Тестирование функциональности;

2. Тестирование удобства использования;

3. Тестирование интерфейса;

4. Тестирование совместимости;

5. Тестирование производительности и скорости загрузки сайта;

6. Тестирование безопасности.

Остановимся поподробнее на первом пункте

«Клиент-сервер» —это вид архитектуры распределенного приложения. В этой архитектуре есть два типа элементов: клиенты и сервер. Они взаимодействуют определенным способом, который называется протоколом. К одному серверу могут подключаться и одновременно запрашивать нужную информацию миллионы клиентов.

Обычно клиенты и сервер расположены на разных вычислительных машинах и взаимодействуют через сеть посредством сетевых протоколов, но они могут быть и на одной машине. Программы-серверы ожидают от клиентских программ запросы и предоставляют им свои ресурсы в виде данных или сервисов, а клиенты инициализируют сеансы с серверами.

Для взаимодействия с веб приложением необходим проколов http.

Протокол HTTP (сокр. от англ. HyperText Transfer Protocol) —протокол прикладного уровня, используемый в веб-приложениях для передачи данных между клиентом и сервером. По умолчанию используется порт 80. HTTP чаще всего использует возможности протокола TCP для пересылки своих сообщений.

HTTPS (сокр. от англ. HyperText Transfer Protocol Secure) —расширение протокола HTTP для поддержки шифрования в целях повышения безопасности. Данные в протоколе HTTPS передаются поверх криптографических протоколов SSL или TLS. HTTPS по умолчанию использует порт 443.HTTP —это так называемый stateless-протокол. Он не имеет состояния, то есть в этом протоколе не существует связи между двумя запросами, которые последовательно выполняются по одному соединению.

Для обхода этого ограничения могут применяться Cookie, которые позволяют использовать сессии с сохранением состояния. Cookie добавляются к рабочему потоку благодаря расширяемости заголовков и позволяют сессии на каждом HTTP-запросе делиться состоянием.

За рендеринг HTML-документов и работу с JavaScript в браузере отвечают специальные компоненты, так называемые движки браузера (browser engine).

На сегодняшний день существует несколько актуальных движков:

* Microsoft Trident – используется в Internet Explorer 11 и 12. Планировалось, что в дальнейшем будет заменен на Blink;
* EdgeHTML – являлся основой для Microsoft Edge, браузера в Windows 10. Сейчас заменён на Blink;
* Gecko – на данный момент используется в Firefox и других продуктах, входящих в Mozilla, например, в почтовом клиенте Thunderbird;
* Blink – наиболее популярный движок, основе браузера с открытым кодом Chromium, на базе которого был разработан браузер Chrome и многочисленные сторонние браузеры: Opera, Яндекс.Браузер, Atom и другие;
* WebKit – движок для браузера Safari, это стандартный браузер в macOS.

## 2.3 Нагрузочное тестирование

Нагрузочное тестирование - важный элемент комплексного тестирования производительности, цель которого - проанализировать скорость отклика системы на внешний запрос. Этот тест позволяет определить, соответствует ли тестируемое приложение или устройство заявленным требованиям. Кроме того, можно понять, как программа отреагирует, если ее используют несколько пользователей одновременно. Основная идея нагрузочного тестирования - создать определенную нагрузку с помощью программного и аппаратного обеспечения, а затем отслеживать индекс производительности продукта.

Основные этапы нагрузочного тестирования:

1. Подготовка — Проводится анализ целей и статистики эксплуатации системы. Определяются бизнес-операции, имеющие значение с точки зрения нагрузки на систему. Создается и согласуется документ «Методика нагрузочного тестирования», который включает: стратегию тестирования, список и описание тестов, критерии успешного завершения, описание средств мониторинга и инструментов нагрузочного тестирования. Осуществляется подготовка тестовых данных, настраивается мониторинг, наполняется база данных.

2. Проведение — Выполняется запись и отладка скриптов нагрузочного тестирования, и реализация сценариев нагрузочного тестирования. С помощью разработанной системы нагрузочного тестирования выполняется тестирование приложений. В рамках нагрузочных испытаний собирается различная статистическая информация, с помощью которой выполняется анализ производительности.

3. Отчет — После проведения нагрузочного тестирования компании клиенту предоставляется отчет, который описывает результаты тестирования, отступления от методики (если имеются), список ошибок, предложения по оптимизации работы системы, общие замечания.

Основными целями нагрузочного тестирования являются:

1. Оценка производительности и работоспособности приложения на этапе разработки и передачи в эксплуатацию

2. Оценка производительности и работоспособности приложения на этапе выпуска новых релизов, патч-сетов

3. Оптимизация производительности приложения, включая настройки серверов и оптимизацию кода

4. Подбор соответствующей для данного приложения аппаратной (программной платформы) и конфигурации сервера

## 2.4 Автоматизированное тестирование

Автоматизированное тестирование ПО — процесс тестирования программного обеспечения, при котором основные функции и шаги теста, такие как запуск, инициализация, выполнение, анализ и выдача результата, производятся автоматически с помощью инструментов для автоматизированного тестирования.

Основной целью автоматизации является оптимизация временных и человеческих ресурсов, затрачиваемых на проведение тестирования. Исключен «человеческий фактор». Сильное достоинство. Все мы люди и никто из нас не застрахован от ошибок. Выполняемый же тест-скрипт не пропустит тест по неосторожности и ничего не напутает в результатах.

* Быстрое выполнение – автоматизированному скрипту не нужно сверяться с инструкциями и документациями.
* Меньшие затраты на поддержку – когда скрипты уже написаны, на их поддержку и анализ результатов требуется, как правило, меньшее время чем на проведение того же объема тестирования вручную.
* Отчеты – автоматически рассылаемые и сохраняемые отчеты о результатах тестирования.
* Выполнение без вмешательства – во время выполнения тестов инженер-тестировщик может заниматься другими полезными делами, или тесты могут выполняться в нерабочее время.

В процессе автоматизации можно выделить три основных этапа:

* начальный этап - этап подготовки и планирования. На этом этапе принимается решение о необходимости автоматизированного тестирования, оцениваются потенциальные возможности и экономический эффект, устанавливаются цели и стратегии автоматизации, а также определяются типы тестов, подходящие для автоматизации. На этом же этапе после тщательного изучения свойств, оценки степени соответствия конкретной задаче и оценки ресурсов, необходимых для поддержания нормальной работы, выбирается инструмент автоматизации. В результате первоначальной разработки устанавливаются требования к описанию тестов, проверяется совместимость средств автоматизации и тестируемого программного обеспечения и тестовой среды, а также разрабатываются точные методы оценки затрат на внедрение. этап активной разработки;
* этап поддержки автоматических тестов. Большая часть времени на этом этапе посвящена описанию, разработке, тестированию и выполнению автоматизированных тестов. Уменьшает количество ресурсов, необходимых для разработки общих функций. В случае крупных проектов команда разработчиков тестов может быть значительно увеличена, а успешно завершенная фаза подготовки и планирования гарантирует минимальные риски. При разработке тестов важно обеспечить возможность автоматического документирования обнаруженных ошибок и составления общего отчета по результатам автоматических тестов. После успешного выполнения автоматических тестов результаты анализируются и при необходимости корректируются. Активная фаза разработки может быть довольно продолжительной, в зависимости от размера проекта.
* этап поддержки автоматических тестов. Для автоматизации тестов выбираются только те тесты, которые тестируют фиксированную часть программы. Однако изменение требований к входу, обновление настроек или структуры среды тестирования может привести к тому, что автоматические тесты будут давать ошибочные результаты. Следовательно, изменения в системе всегда следует отслеживать, и при необходимости автоматизированные тесты следует адаптировать или модифицировать, чтобы поддерживать их в актуальном состоянии.

# 3 ТЕСТИРОВАНИЕ МОБИЛЬНЫХ ПРИЛОЖЕНИЙ

Очевидно, что мобильное приложение сильно отличается от веб-версии. Поэтому необходимо учитывать это при планировании процесса тестирования.

Рассмотрим основные различия между мобильными и веб-версии:

* Мобильное устройство – это система, которая не обладает мощными характеристиками. Таким образом, он не может работать как персональный компьютер.
* Тестирование мобильных приложений проводится на мобильных телефонах (Apple, Samsung, Nokia), в то время как веб-версия тестируется на центральном процессоре.
* У мобильных устройств бывают разные разрешения. Размер экрана мобильного телефона меньше, чем у веб-версий.
* Выполнение и прием вызовов является основной задачей телефона, поэтому приложение не должно вмешиваться в эту важную функцию.
* Широкий спектр конкретных операционных систем и компонентных конфигураций: Android, iOS, BlackBerry.
* Операционная система мобильного телефона быстро устаревает.
* Мобильные устройства используют сетевые подключения (3G, 4G, Wi-Fi), широкополосное подключение к настольному ПК или Wi-Fi.
* Мобильные устройства постоянно осуществляют поиск сети. Вот почему необходимо протестировать приложение с разной скоростью передачи данных.
* Инструменты, которые хорошо подходят для тестирования веб приложений, не полностью подходят для тестирования мобильных приложений.
* Мобильные приложения должны поддерживать несколько входных каналов (клавиатура, голос, жесты и т. д.), мультимедийные технологии и другие функции, повышающие их удобство использования.

Тестирование мобильных приложений можно разделить на ручное и автоматизированное. Необходимо понимать, что в краткосрочной перспективе выгоднее использовать ручное, но в долгосрочной - автоматизированное.

Конечно, нет однозначных ответов на то, какую стратегию лучше всего выбрать. Однако сочетание различных вариантов наиболее оптимально. Например, вы можете использовать симуляторы на самых ранних этапах вашего тестирования. Но лучше использовать реальные устройства (физические или облачные) на заключительных этапах. Автоматизированное тестирование предпочтительнее для нагрузочного и регрессионного тестирований.

## 3.1 Ручное тестирование

Самый большой из недостатков ручного тестирования — человеческий фактор. Он, конечно, влияет на всё, происходящее в команде — и на работу участников, и на события, происходящие на стороне клиента. В случае QA причиной того, что тестировщик будет слабо погружен в процесс и пропустит потенциальную ошибку может стать всё что угодно — недостаток опыта, семейные проблемы или головная боль.

Один и тот же сценарий два тестировщика могут проверить разными способами. Что с этим делать? Важно, чтобы каждый непредусмотренный или отличающийся от ожидаемого результат был зафиксирован в виде кейса. Чтобы любой тестировщик мог проверить его, совершив тот же набор действий. Но и этого может быть мало — если кейс окажется недостаточно подробным, а тестировщик — не разберётся в описании. Гарантировать стопроцентное исключение человеческого фактора, конечно, невозможно — но можно постараться максимально снизить вероятность проблем, которые он вызывает.

Это тоже негативно сказывается на сроках поставки фичи в продакшн и трудозатратах: ведь теперь к периодически проводимым базовым кейсам и регрессии добавляются и “хитрые” кейсы, придуманные тестировщиками в процессе.

Усугубляет ситуацию вероятность того, что часть встреченных багов ещё не будет иметь строгого описания, потому что причины их возникновения пока не понятны. Как бороться с такими повторными тестированиями? Либо найти уже источник ошибки и устранить его, либо — всё-таки автоматизировать прохождение проблемных кейсов — в этом случае переход к программным тестам будет вполне оправдан как с точки зрения времени, так и финансово. (Нет, это не противоречит вышесказанному — потому что такие ситуации возникают уже в ходе активной разработки и к этому времени вы уже в любом случае развернёте процессы автотестирования).

В любом случае — появление первых кейсов, нуждающихся в регрессивных тестах или релиз второй версии приложения и соответствующее этим событиям масштабирование команды — тот момент, когда автоматизация станет актуальна (но не исключит ручное тестирование новых возможностей). На этом этапе автоматизация уже начнёт экономить время: разработчик сможет сам, ещё до передачи QA-отделу запустить регресс-тесты новой фичи, чтобы убедиться, что она не ломает существующий функционал, а тестировщику не придётся лишний раз проходить по набившим оскомину базовым кейсам. Ещё одно преимущество внедрения автотестов на этом этапе — их можно запускать по определённому расписанию — каждую ночь, в дни окончания спринтов или при публикации новых сборок приложения.

При этом нельзя забывать несколько вещей:

* создание кейсов и написание автотестов будет требовать времени — закладывайте его в спринты;
* убедитесь, что кейс автотеста прописан хорошо и подробно и описывает возможную проблему или существующий сценарий во всей полноте;
* проверьте, правильно ли работает автотест и действительно ли он проверяет то, что нужно и делает это качественно.

Достоинства ручного тестирования мобильных приложений:

1. Это более экономически выгодно в краткосрочной перспективе.
2. Ручное тестирование более гибкое.
3. Лучшее моделирование действий пользователя.
4. Недостатки ручного тестирования мобильных приложений:
5. Ручные тестовые примеры трудно использовать повторно.
6. Менее эффективно выполнение определенной постоянной задачи.
7. Процесс тестирования медленный.
8. Некоторые типы тестовых случаев не могут быть выполнены вручную (нагрузочное тестирование).

## 3.2 Автоматизированное тестирование

Автоматизация тестирования помогает решить сразу несколько проблем — в том числе если речь идёт о мобильных приложениях. Вместо того чтобы вручную проводить рутинные трудоемкие процедуры, специалисты могут делегировать значительную их часть фреймворкам. Автоматизация упрощает проверку и помогает ускорить регрессионное тестирование, а также даёт возможность использовать ранее недоступные типы тестирования.

В идеале вы должны настроить как можно больше кейсов, что позволит вам автоматизировать около 80% ваших процессов тестирования. Существуют конкретные тестовые случаи, которые должны быть автоматизированы, список перед вами:

* автоматизируйте самые частотные тест-кейсы;
* автоматизируйте тестовые случаи, которые легко автоматизировать;
* автоматизируйте тест-кейсы, которые имеют предсказуемые результаты;
* автоматизируйте самые утомительные ручные тесты;
* автоматизируйте тест-кейсы, которые невозможно выполнить вручную;
* автоматизируйте тест-кейсы, которые выполняются на нескольких различных аппаратных или программных платформах и конфигурации;
* автоматизируйте часто используемые функции.

Преимущества автоматизированного тестирования приложений:

* Процесс тестирования занимает мало времени.
* Экономичность в долгосрочной перспективе использования.
* Автоматизированные тестовые случаи легко использовать повторно.
* Единственное решение для некоторых видов тестирования (тестирование производительности).
* Результаты испытаний легко доступны.
* Недостатки автоматизированного тестирования приложений:
* У некоторых мобильных средств тестирования есть ограничения.
* Процесс тестирования занимает много времени.

Автоматизированное тестирование наименее эффективно в определении удобства пользования, что подводит нас к следующем разделу.

# 4 UX ТЕСТИРОВАНИЕ

Юзабилити-тестирование – это сценарный план, предложенный специалистом на основе анализа вашего ресурса или виртуального бюро услуг. Цель юзабилити-анализа – экспериментальным путем установить, удобен ли ресурс для клиентов, а если нет – вывести парадигму требуемых улучшений.

Только на основе юзабилити-тестирование вы сможете понять – какие исходные данные нужно использовать для A/B-тестирования (сравнительного анализа существующей и экспериментальной страницы, на который изменен один или ряд фрагментов).

Определив группу задач, которые будут решаться в ходе юзабилити-тестирования, вы даете возможность потенциальным клиентам озвучить свои недовольства и пожелания.

Базовый набор инструментов юзабилити-тестирования включает в себя:

* аналитические системы;
* web-визор;
* эвристический анализ.

К примеру, в процессе юзабилити-тестирования мобильных приложений (такая потребность чаще встает перед маленькими предприятиями), вам желательно заручиться достаточным количеством пользовательских отзывов. Это очень важно, ведь вам нужен ответ на вопрос: почему часть клиентов ушла к другим продавцам, не совершив какие-либо действия именно на вашем ресурсе.

В зависимости от поставленных вами задач по улучшению конверсии сайта и особенностей вашей желаемой целевой аудитории, можно провести:

* классическое юзабилити-тестирование;
* онлайн-юзабилити-тестирование.

Кроме того, вам не составит труда понять, – как проводить юзабилити-тестирование. Последовательность ваших действий будет примерно такой:

* выбор сервиса автоматизации (на европейском рынке пользователи обычно предпочитают usertesting.com);
* выбор фокус-группы (к примеру, 20 девушек возрастом от 18 до 28 лет, регулярно заказывающих в интернет-магазинах аксессуары для гаджетов);
* проведение юзабилити-тестирования в течение нескольких дней посредством установки у пользователей софта для захвата и записи экрана.

Хорошо, если разработчики вашего ресурса убедили вас провести юзабилити-тестирование еще до того, как вы окунулись в мир реальных продаж. В этом случае изначально можно быть уверенным в том, что посадочная страница либо страницы каталога, формы выбора товара и оплаты соответствуют устоявшимся представлениям клиентов об удобстве пользования.

Провести тестирование можно и самостоятельно, здесь важен масштаб задач, которые вы перед собой ставите. В качестве фокус-группы, проводя юзабилити-тестирование самостоятельно, вы можете использовать друзей, френдов из соцсетей, людей, зарегистрированных на бирже фрилансеров, однако такой подбор пользователей оправдает себя лишь в случае товаров и услуг, нужных всем без исключения: к примеру, простейших парикмахерских услуг. Но, если вы занимаетесь сопровождением деловых конференций или продаете автозапчасти для салонного транспорта, вам понадобятся реальные потребители ваших услуг, часто прибегающие к механизмам онлайн-покупок и имеющие опыт сотрудничества с разными продавцами.

Следовательно, обратившись к специалистам, вы сможете быть уверенными в том, что для вашего ресурса подберут тематических клиентов либо профессиональных юзабилистов.

Еще один плюс проведения юзабилити-тестирования узкопрофильной web-компанией: по факту завершения анализа вам предоставят не просто отчет о выявленных ошибках, а пакет рекомендаций по улучшению юзабилити ресурса.

Итак, повторим: для тестирования интернет-магазина детских товаров понадобятся молодые родители с доходами не выше среднего, музыкальных дисков – меломаны, готовые платить за лицензионную продукцию, а услуг солярия – жильцы микрорайона, в котором находится ваш салон.

Нельзя ограничивать фокус-группу небольшим количеством пользователей: безусловно, грубые ошибки помогут установить и первые же пять человек, однако глубинные недочеты могут укрыться от их внимания.

Еще одно преимущество курирования процесса профессионалом – специалист точно знает, как проводить юзабилити-тестирование, иначе говоря, он верно поставит задачи и распределит их последовательность.

Прежде чем юзабилити-тестирование будет запущено, необходимо провести подготовительную работу:

* вычленить KPI (ключевые показатели эффективности) вашего ресурса: как правило, это оформление заказа, запрос на обратный звонок, подписка на рассылку;
* вычленение слабых звеньев (например, страниц, с которых пользователь, открыв, сразу уходит) и «холодных этапов», когда, например, пользователь уходит с сайта, дойдя до фильтра товаров;
* написание сценарного плана тестирования (к примеру, составление модулей предполагаемых действий пользователя: «посмотреть каталог – заказать звонок», «посмотреть каталог – отложить в корзину», «посмотреть каталог – оформить покупку»);
* формулирование ТЗ для пользователя фокус-группы, к примеру, «Вы – девушка 20 лет, ищите оригинальный подарок для шефа своей компании стоимостью от столько-то до столько-то».

Как показывает практика, по итогам юзабилити-тестирования могут вскрыться совершенно неожиданные для владельца ресурса причины непопулярности сайта или интернет-магазина у целевой группы. К примеру, некоторое количество пользователей обескураживает и отталкивает необходимостью пройти многоступенчатую регистрацию, прежде чем оформить покупку. Иногда пользователь прерывает процесс выбора товара, столкнувшись с обилием полей, которые необходимо заполнить для перехода в нужный сегмент каталога.

И самое важное: организация юзабилити-тестирования должна охватить и выявить как можно больше проблемных зон ресурса. Лишь в этом случае у специалиста будет достаточное количество исходных данных для составления для вас указаний по улучшению юзабилити.

# 5 МЕНЕДЖМЕНТ ТЕСТИРОВАНИЯ

## 5.1 Управление тестированием

Для корректного взаимодействия между членами команды тестировщиков и организации грамотного тестирования продукта, используется единая “СУТ” (Система управления тестированием). Системы управления тестированием обычно используются для планирования ручного тестирования, сбора данных о результатах прохождения чек-листов и тест-кейсов, а также для получения оперативной информации в виде отчетов. Системы управления тестированием помогают оптимизировать процесс тестирования и обеспечивают быстрый доступ к анализу данных, средствам совместной работы и более качественному взаимодействию между несколькими проектными группами.

После старта тестирования проекта члены команды могут взаимодействовать через одну из систем управления тестирования путём создания тест-кейсов, чек-листов, назначая ответственных за прохождение их лиц, что упрощает и улучшает качество взаимодействия лиц, проводящих тестирование в рамках конкретного проекта. При создании или прохождении тестов и чек-листов пользователи могут получить доступ к различным функциям систем управления тестированием, которые автоматизируют данную деятельность и благотворно влияют на скорость и качество её выполнения.

## 5.2 Бизнес-ценность тестирования

На сегодняшний момент сложно отрицать ценность специалистов по тестированию. Основная задача данного специалиста - предотвратить дефекты и, следовательно, обеспечить высокое качество процесса разработки и его результатов. Для более детального рассмотрения функций составим список задач специалиста по тестированию продукта.

* Выявление слабых мест и несоответствия в продукте на всех этапах разработки.
* Помощь в определении требований к проекту.
* Предоставление исчерпывающей информации о качестве продукта.
* Тестирование продукта на протяжении всех фаз жизненного цикла разработки системы (software development lifecycle, SDLC).

Важно отметить, что специалисты по тестированию заинтересованы в том, чтобы сделать любой продукт удобным для пользователя как в плане функциональности, так и в плане дизайна. Для этого они тесно взаимодействуют со всеми членами команды и постоянно обращаются к заданным требованиям.

Для описания непосредственного влияния тестирования на “Бизнес-сторону” проекта рассмотрим конкретные примеры, где присутствие тестировщика в команде экономически обусловлено:

* Безопасный бизнес. У вас есть платёжная система, и она работает нормально. Пользователь платит за услугу и получает её. Однако вы не проверили все возможные случаи, и деньги идут не вам, а на случайный банковский счёт. Такой недочёт может очень дорого обойтись.
* Экономия денег. На приведённой ниже диаграмме хорошо видна взаимосвязь между этапами жизненного цикла и затратами. Гораздо дороже исправить ошибку, чем предотвратить её. Исправление одной ошибки может повлечь за собой другие, поэтому количество ваших проблем будет быстро увеличиваться.
* Защита репутации. Если выпустить багованный продукт и пользователи не будут довольны работой с ним, в дальнейшем их будет сложно убедить, что проблема решена и они могут снова им пользоваться. Первое впечатление сложно изменить, поэтому предоставьте качественный продукт. Если он не был протестирован вдоль и поперёк, то продукт может работать неправильно или не работать вовсе. Тестирование требует теоретических знаний, поэтому будет сложно обеспечить качество, если вы не профессионал.
* Контроль процесса. Если процесс разработки не контролируется и идёт вразрез с установленными требованиями, итоговый продукт может сильно отличаться от запланированного.

## 5.3 Управление дефектами

Для управления дефектами применяют баг-трекинговые системы, которые упрощают запись и ведение информации о дефектах.

При обнаружении дефект проходит следующий жизненный цикл:

* Дефект открыт, когда его обнаружили и создали отчёт о нем.
* Дефект в работе, когда он назначен на разработчика и разработчик начал работу над ним.
* Дефект исправлен, когда разработчик внес изменения (исправление или fix) в программный код и сохранил эти изменения в общем репозитории.
* Дефект закрыт, когда было проверено, что он действительно исправлен в новой версии продукта.
* Дефект открыт заново в случае, если ранее исправленный дефект снова был обнаружен в системе.

# 6 ОТЧЕТ ПО ТЕСТИРОВАНИЮ ПРОЕКТА

## 6.1 Описание проекта

Целью данного проекта является тестирование экрана сообщений и чатов в мобильном приложении «Юла». Необходимо протестировать их, составить чек лист проверок и тест-кейсы.

## 6.2 Чек-лист

Ниже приведены выполненные чек-листы:

* Проверка перехода назад/другой экран
* Проверка отображения заглушки "Нет сообщений"
* Отображение заглушки "Отсутствует подключения”
* Отображение отправленного сообщения
* Отображение входящего сообщения
* Скролл списка сообщений
* Удаление сообщения
* Обновление экрана "Сообщения"
* Переход на экран Сообщения"
* Переход на другой экран
* Переход ‘назад“
* Верстка экрана "Сообщения": нет сообщений
* Отображение заглушки “Отсутствует подключение”
* Тап “Попробовать еще раз" отсутствует подключение к интернету
* Отображение отправленного сообщения (короткое)
* Отображение отправленного сообщения (длинное)
* Отображение отправленного сообщения: прочитано
* Отображение иконки верифицированного юзера в списке сообщений
* Отображение входящего текстового сообщения
* Отображение входящего сообщения с изображением.
* Отображение входящего сообщения: скрывается под многоточие.
* Скролл списка сообщений
* Обновление экрана "Сообщения"
* Переход в сообщение
* Переход на экран "Сообщения из чата”.

## 6.3 Тест-кейсы

В таблице 6.1 содержатся тест-кейсы для приложения экрана сообщений

Таблица 6.1 – Тест-кейсы для экрана сообщений

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№ теста** | **Описание** | **Начальные условия** | **Шаги** | **Ожидаемый результат** |
| 1 | Правильное отображение раздела “Сообщения” | Приложение установлено, но не запущено | 1. Запустить приложение  2. Открыть пункт меню “Сообщения” | Корректное отображение раздела “Сообщения” |
| 2 | Обновление списка сообщений | Открыт раздел “Сообщения” | 1. Свайп вниз | Обновление произошло |
| 3 | Переход на пункт меню “Главная страницу” | Открыт раздел “Сообщения” | 1. Нажатие на иконку “Главная страница” | Произошел переход на страницу “Главная страница” |
| 4 | Переход на пункт меню “Поиск” | Открыт раздел “Сообщения” | 1)Нажать на пункт меню “Поиск” | Произошел переход на страницу “Сообщения” |
| 5 | Переход на пункт меню “Добавление объявления” | Открыт раздел “Сообщения” | 1. Нажать иконку “Добавления сообщений” | Произошел переход на страницу добавления объявления |
| 6 | Переход на пункт меню “Сообщения” | Открыт раздел “Сообщения” | 1. Нажатие на иконку “Сообщения” | Остаемся на исходной позиции |
| 7 | Переход на пункт меню “Избранное” | Открыт раздел “Сообщения” | 1. Нажатие на иконку “Избранное” | Произошел переход на страницу “Избранное” |
| 8 | Открытие чата с собеседником | Открыт раздел “Сообщения” | 1. Выбрать нужный диалог 2. Нажать на поле чата | Открытие экрана чата |
| 9 | Удаление чата с собеседником | Открыт раздел “Сообщения” | 1. Выбрать нужный диалог 2. Свайпнуть диалог влево 3. Нажать на кнопку удалить | Удаление чата |

Таблица 6.2 – Тест-кейсы для экрана чата

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№ теста** | **Описание** | **Начальные условия** | **Шаги** | **Ожидаемый результат** |
| 1 | Переход на предыдущую страницу | Открытый экран чата с собеседником | 1. Нажатие на кнопку “Назад” | Отображение раздела “Сообщения” |
| 2 | Нажатие на кнопку профиля | Открытый экран чата с собеседником | 1. Нажать на иконку с фотографией или стрелочкой возле нее | Открытие выпадающего списка с доступными функциями |
| 3 | Открытие профиля собеседника | Открытый экран чата с собеседником | 1. Нажать на иконку с фотографией или стрелочкой возле нее 2. Нажать на кнопку “Открыть профиль” | Переход на страницу профиля собеседника |
| 4 | Нажатие на кнопку “Оставить отзыв” | Открытый экран чата с собеседником | 1. Нажать на иконку с фотографией или стрелочкой возле нее 2. Выбрать из предложенных вариантов приобрел ли товар покупатель | Переход в меню выставления оценки |
| 5 | Обращение в поддержку | Открытый экран чата с собеседником | 1. Нажать на иконку с фотографией или стрелочкой возле нее 2. Нажать на кнопку “Написать в поддержку” | Переход на страницу обратной связи |
| 6 | Пожаловаться на пользователя | Открытый экран чата с собеседником | 1. Нажать на иконку с фотографией или стрелочкой возле нее 2. Нажать на кнопку “Пожаловаться” | Переход на страницу “Отправить жалобу” |
| 7 | Удаление чата | Открытый экран чата с собеседником | 1. Нажать на иконку с фотографией или стрелочкой возле нее 2. Нажать на кнопку “Удалить чат” 3. Подтвердить удаление | Чат удален |
| 8 | Заблокировать  пользователя | Открытый экран чата с собеседником | 1. Нажать на иконку с фотографией или стрелочкой возле нее   2) Нажать на кнопку “Заблокировать” | Пользователь заблокирован |
| 9 | Аудиозвонок | Открытый экран чата с собеседником | 1. Нажать на иконку с рисунком телефона 2. Выбор как совершить звонок (по номеру телефона или через Юлу) | Совершается звонок собеседнику |
| 10 | Видеозвонок | Открытый экран чата с собеседником | 1. Нажать на иконку с рисунком камеры | Совершается видеозвонок собеседнику |
| 11 | Копирование сообщения | Открытый экран чата с собеседником | 1. Зажать сообщение 2. Выбор пункта “Скопировать” | Сообщение скопировано |
| 12 | Пожаловаться на спам | Открытый экран чата с собеседником | 1. Зажать сообщение собеседника 2. Выбор пункта “Пожаловаться на спам” | Жалоба на сообщение отправлена |
| 13 | Прикрепить новую фотографию | Открытый экран чата с собеседником | 1. Нажать на кнопку “Прикрепить” 2. Выбор пункта “Сделать фотографию” 3. Сделать фотографию | Фотография отправлена |
| 14 | Прикрепить фотографию из галереи | Открытый экран чата с собеседником | 1. Нажать на кнопку “Прикрепить файлы” 2. Выбрать пункт “Прикрепить фотографию из галереи” 3. Отметить нужные фотографии 4. Нажать “Выбрать фото” | Фотография прикреплена |
| 15 | Отправка сообщения | Открытый экран чата с собеседником | 1. Нажать на поле для ввода текста 2. Ввести текст 3. Нажать на кнопку отправки сообщения | Сообщение отправлено |
| 16 | Просмотр товара | Открытый экран чата с собеседником | 1. Нажать на прикрепленный товар | Переход на страницу товара |

## 6.4 Найденные дефекты

Фактический результат соответствует ожидаемому во всех случаях, кроме удаления чата. Найденный дефект написан ниже.

Найденный дефекты: Удаление чата не удаляет историю сообщения, при повторной отправке сообщения появляется история.

# ЗАКЛЮЧЕНИЕ

В ходе выполнения работ были приобретены базовые знания о тестировании. Были изучены термины, рассмотрены инструменты, применяющиеся в данной сфере. Команда приобрела и углубила знания в области функционирования мобильных приложений на различных операционных системах. Были рассмотрены методы тестирования, способы ведения учета о дефектах.

Была проанализирована работа систем управления тестированием (менеджмента тестирования), которые используются для хранения информации, для получения информации в виде отчета о стадии тестирования и качестве тестируемого продукта. Следует отметить также высокую бизнес-ценность тестирования в современном профессиональном мире и существенную важность управления дефектами.

Выполняя практические задачи по тестированию, команда получила практические навыки тестирования мобильных приложений, составления тестовой документации.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Борис Бейзер «Тестирование черного ящика» от 04.01.2004 под редакцией издательства «Питер» https://codernet.ru/books/QA/testirovanie\_chernogo\_yashchika\_boris\_beyzer/
2. Гленфорд Майерс, Том Баджетт, Кори Сандлер «Искусство тестирования программ» от 22.10.2009 под редакцией издательства "Финансы и статистика"
3. Канер Сэм, Фолк Джек, Нгуен Енг Кек «Тестирование программного обеспечения» от 13.05.2001 под редакцией издательства «ДиаСофт». https://codernet.ru/books/QA/testirovanie\_programmnogo\_obespecheniya\_sem\_kaner\_2001/
4. Лиза Криспин, Джанет Грегори «Гибкое тестирование» от 12.11.2016 под редакцией издательства «Signature Series»
5. Рекс Блэк «Ключевые процессы тестирования» от 17.04.2014 под редакцией издательства «Лори» https://codernet.ru/books/QA/klyuchevye\_protsessy\_testirovaniya\_reks\_blek/
6. Роберт Калбертсон, Крис Браун, Гэри Кобб — «Быстрое тестирование» от 16.02.2007 под редакцией издательства «Вильямс» https://codernet.ru/books/QA/bystroe\_testirovanie\_robert\_kalbertson/
7. Святослав Куликов «Тестирование программного обеспечения. Базовый курс» от 29.09.2015 под редакцией «Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International» https://fktpm.ru/file/113-svyatoslav-kulikov-testirovanie-po-bazovyi-kurs.pdf
8. Рекс Блэк «Ключевые процессы тестирования» от 17.04.2014 под редакцией издательства «Лори» https://codernet.ru/books/QA/klyuchevye\_protsessy\_testirovaniya\_reks\_blek/
9. Список обучающих лекций по работе над тестирование под редакцией «Lime Lab»