# Письменные ответы на вопросы

1.Если на составление тестовой документации и на само тестирование имеется очень много свободного времени, тогда почему бы не применить таблицу принятия решений. Количество комбинаций 2\*5\*9\*2\*2=360 (всего то!!!).

Естественно целесообразно применить технику попарного тестирования (воспользоваться ресурсом <https://pairwise.teremokgames.com/>) и выбрать уникальные пары, с целью сокращения времени и уменьшения количества тестов.

2.В тест-кейсе не хватает важного элемента - **уникального идентификатора** тест-кейса.

Я бы слегка видоизменил **заголовок**. Вместо “Отображение главной страницы Яндекс Практикума для неавторизованного пользователя” написал: “Главная страница Яндекс Практикума для неавторизованного пользователя отображается корректно”.

В **предусловии** заменил бы текст на “открыть новое окно в режиме инкогнито”

**Шаги.** Убрал бы излишнюю детализацию и фамильярность. Оставил бы такой вариант: “Открыть главную страницу [https://practicum.yandex](https://practicum.yandex.ru/).ru/”

**Ожидаемый результат** тоже бы заменил на ““Главная страница отображается корректно”

Данные изменения добавляют в проверку большую конкретику и превратят кейс в хороший тест-кейс.

3.В случае возникновения подобной ошибки мои действия:

запустить DevTool (просмотреть код).

Обновить страницу.

Перейти на вкладку Network.

Произвести на странице манипуляцию с чек-боксом.

На вкладке Network найти нужные request и response с кодом ошибки 4хх, ошибка на стороне фронта (несколько типовых ошибок 400 - Bad Request, 403 - Доступ запрещен, 404 - Ресурс не найден) или 5хх, ошибка на стороне бека (Ошибка 500, или [Internal Server Error](https://www.google.com/search?sca_esv=5f6572bf4e7c5530&sxsrf=AE3TifPoVK8HcFQTUKLSw7thG4vc-97IBw%3A1754297652011&q=Internal+Server+Error&sa=X&ved=2ahUKEwjKoYOD5PCOAxVPGxAIHUoDJ34QxccNegQIMRAB&mstk=AUtExfBHPKSKXWRuGR3tOcs7IjfeLF9D-aiBvYOt1a_yD_t5sjlS0dcb13UZtQR9C7isW28js6CYTV8w2A2nA6vtuy-0dTE3gzqHvZJImw09YGebwIw0e-v7SJH5S2hPSeTCF2tAGk8XAZoGOncM0u_CqPnJItEaNq7ILpdmstAgO2yUe8zryRDfwE4cnTEEtMt04bQ9wgtjlIa_ds6r1YgSqi-hTS_AjTYw30m6guoR3L4J_giYX_nnEKMKfAEcuGuwOn3VmLMgClEljTnRp9I5Knnp&csui=3), 502 - Ошибка шлюза и т.д.).

4.Метод Get предназначен для запросов на чтение данных (можно вывести, например, таблицу имеющихся товаров или все заказы гражданина Иванова).

Для удаления потребуется метод Delete.

Как показал недавний опыт из командного спринта, при тестировании Api запрос Delete вел себя как Get (вместо удаления он выводил данные в качестве ответа или же выдавал ошибку “метод Get не разрешен”. На что были составлены Баг репорты.

Использование GET для удаления данных может привести к путанице, ошибкам в работе приложения и проблемам с безопасностью.

Нарушение семантики HTTP:  
Изменение состояния сервера (например, удаление данных) не является его задачей. Потенциальные проблемы с безопасностью:  
GET-запросы часто кэшируются браузерами и прокси-серверами, что может привести к нежелательному удалению данных, если кто-то получит доступ к кешированной версии запроса.

Проблемы с отладкой и поддержкой:  
Использование GET для удаления данных затрудняет понимание и поддержку кода, так как логика работы приложения становится менее очевидной.

Несоблюдение [идемпотентности](https://www.google.com/search?cs=0&sca_esv=5f6572bf4e7c5530&sxsrf=AE3TifOO_igMfQJcbhkxunCKKOC6ALaaPw%3A1754296567852&q=%D0%B8%D0%B4%D0%B5%D0%BC%D0%BF%D0%BE%D1%82%D0%B5%D0%BD%D1%82%D0%BD%D0%BE%D1%81%D1%82%D0%B8&sa=X&ved=2ahUKEwjDxOT93_COAxXoGxAIHQ19OtwQxccNegQIEBAB&mstk=AUtExfBufIxyz7Ne5c_O-qYVu4uCFlDVkfbJ7BupgzSLJQG9lZ4_ttCgGm9JamDICviM7MigHmwhalXDW27DnwbtdgMwjEJb0lfMxx4KJxoNUCk21m4rpd1N1Q8O0mGExxZ_NnovOhL0KqHqmnForOIgoVtO6VlQz4Pt1ehZC7959yqxPZ9E_yaK3ntXLsS1iVMAQi_GseKK0ltzIfzVn_7rjefSYQZ_IwoGJurzoh41mu5n9IVeHTZrQG9FXALoOyic0gasS0DUy0xitBVLGFDhUye-&csui=3):  
Метод GET, как правило, предполагается идемпотентным, то есть его повторное выполнение не должно изменять состояние сервера. Удаление данных при каждом GET-запросе нарушает это правило

5.В таблице сотрудников (employee) первичный ключ (делающий запись уникальной) - id — идентификатор сотрудника, публичный ключ (для связи с данными из других таблиц) - position\_id — идентификатор должности.

В таблице должностей (position) первичный ключ - id — идентификатор должности.

Так как у нас самая типовая связь “один ко многим”, то этого будет достаточно.

6.Данный запрос требует слияния данных таблиц (например, с помощью INNER JOIN). Он будет составлен следующим образом:

SELECT e.fio,

p.name,

p.salary

FROM position AS p

INNER JOIN employee AS e ON p.id = e.position\_id;

7. В автотесте не хватает команды проверки assert. Она работает как логическое выражение: возвращает True или False. В нашем примере, если результат целочисленного деления правда 2, то вернет True, а иначе — False.

def test\_integer\_division():

a = 5//2;

assert a == 2

test\_integer\_division();

8.Классы эквивалентности существуют двух видов: набор (нельзя упорядочить и сравнить) и диапазон (можно сравнить и упорядочить по убыванию или возрастанию).

У набора нет граничных значений. Например это строка только из символов кириллицы в нижнем регистре или строка со специальными символами.

А диапазон содержит первую и последнюю точки (если не уходит в положительную или отрицательную бесконечности). Граничные значения и их соседи (шаг внутрь и наружу) неотъемлемая часть диапазона, на которых очень часто могут возникать ошибки.

9.Если мы проверяем тип диапазон мы обязаны рассмотреть точки на границах и соседей (6 значений, могут быть не всегда), а также взять еще одну проверку внутри диапазона. Из получившихся семи проверок можем исключить дубли из других диапазонов. Убрать проверку внутри класса нельзя, так как она является контрольной для всего класса эквивалентности.

10.Для воспроизведения бага постараюсь записать работу приложения до момента прекращения его корректной работы (зная шаги приведшие к ошибке, запись не будет слишком объемной). Для передачи логов с мобильного устройства воспользуюсь командной строкой на компьютере (adb logcat), предварительно связав его с устройством или через USB или Wi-Fi.

11. После исправления первое, что нужно провести - ретест багов и санитарное тестирование, чтобы убедиться, что эти изменения работают как ожидалось, и что они не нарушили основные функции системы.

А дальше будет развилка (достаточно ли у нас времени).

Времени достаточно:

Смоук тестирование, проверка критически важных функциональностей, которая критически важна для взаимодействия между пользователем и приложением (протестировать, открывается ли приложение, полностью ли загружается интерфейс, отвечает ли база данных).

Полирнуть сверху еще и регрессом, чтобы наверняка все работало.

Времени ограничено:

ограничимся только дымовым тестированием.

12. Если на запись было произнесено, что http язык, то прошу прощения, это явная оговорка. HTTP - протокол передачи информации. HTML (на видео подразумевался он) - язык гипертекстовой разметки текста, может быть использован для составления тела запроса, как и XML (например).

SOAP используется WSDL (Web Services Description Language), который является XML-основанным языком.

REST не имеет единого стандартного языка для определения сервисов.

13.Чем сложнее проект/приложение, тем целесообразнее писать тест-кейсы (например медицинское оборудование и где цена ошибки это здоровье или жизнь). А вот если продукт на стадии доработок или постоянных изменений, то чек-лист (его будет проще поддерживать).

Время. Если его достаточно, то можно прибегнуть к тест-кейсам. В противном случае - чек-лист, но с потенциальной перспективой доработки до тест-кейса.

Команда. В некоторых компаниях может быть изначально прописано какую документацию применять. В противном случае все зависит от количества людей в команде, их опыта.

Главный итог: соблюдение баланса, чтобы тестирование не превратилось только в работу с документацией.

# Отчет о тестировании

## Функциональное тестирование веб-приложения

Приложение проверено на стенде (<https://e9f6c982-1406-49c2-9686-1cd7b6eef3cb.serverhub.praktikum-services.ru/>). Все известные требования были покрыты чек-листом: (<https://docs.google.com/spreadsheets/d/1uncyVvKie50oKX-Z5LdlOhmu78eGzJS6HP2gaPlkgSY/edit?usp=sharing>).

Результаты выполнения тестов можно посмотреть здесь: (<https://docs.google.com/spreadsheets/d/1uncyVvKie50oKX-Z5LdlOhmu78eGzJS6HP2gaPlkgSY/edit?usp=sharing>). Из (202) успешно прошло (173), не прошло — (22), пропущено (6) и заблокировано (1).

Список багов, найденных при тестировании, разбит по приоритетам:

1. Блокирующие:  
[<https://docs.google.com/spreadsheets/d/1uncyVvKie50oKX-Z5LdlOhmu78eGzJS6HP2gaPlkgSY/edit?usp=sharing>]

2. Критичные:  
[<https://docs.google.com/spreadsheets/d/1uncyVvKie50oKX-Z5LdlOhmu78eGzJS6HP2gaPlkgSY/edit?usp=sharing>]

3. Высокий приоритет:  
[<https://docs.google.com/spreadsheets/d/1uncyVvKie50oKX-Z5LdlOhmu78eGzJS6HP2gaPlkgSY/edit?usp=sharing>]

4. Низкий приоритет:  
[<https://docs.google.com/spreadsheets/d/1uncyVvKie50oKX-Z5LdlOhmu78eGzJS6HP2gaPlkgSY/edit?usp=sharing>]

5. Тривиальные:  
[<https://docs.google.com/spreadsheets/d/1uncyVvKie50oKX-Z5LdlOhmu78eGzJS6HP2gaPlkgSY/edit?usp=sharing>]

Заключение:

1. Какой баг показался самым критичным?

Br\_lgren\_38 и Br\_lgren\_41 не позволяют оформить заказ, так как вместо ожидаемого оформления, появляется дополнительное всплывающее окно, а зачем и для чего в требованиях не указано. И к тому же с него оформить заказ тоже нельзя. (https://docs.google.com/spreadsheets/d/1uncyVvKie50oKX-Z5LdlOhmu78eGzJS6HP2gaPlkgSY/edit?gid=1186534874#gid=1186534874&range=B16)

1. На твой взгляд, какая самая «хитрая» серая зона есть в требованиях?

Пользователь может выбрать другую дату для поля Когда привезти самокат. Даже из прошлого и далекого будущего. Пояснений нет.

1. Проверенная тобой функциональность готова к релизу? Почему?

Пока не исправлены блокирующие и критические баги, а также ряд ошибок с правильным наименованием кнопок, плейсхолдеров, релиз такого продукта должен быть перенесен.

## 

## Ретест багов в мобильном приложении

Был проверен фикс багов. Из них не исправлено (1), исправлено — (3). В ходе фикса багов было обнаружено новых (1).

Список багов можно посмотреть здесь: [<https://docs.google.com/spreadsheets/d/1uncyVvKie50oKX-Z5LdlOhmu78eGzJS6HP2gaPlkgSY/edit?usp=sharing>].

## Регрессионное тестирование мобильного приложения по готовым тест-кейсам

Результаты выполнения регрессионных тестов можно посмотреть здесь: (<https://docs.google.com/spreadsheets/d/1uncyVvKie50oKX-Z5LdlOhmu78eGzJS6HP2gaPlkgSY/edit?usp=sharing>).

Из (10 тестов) успешно прошло (2), не прошло — (7), заблокировано - (1).

Список багов, найденных при тестировании, разбит по приоритетам:

1. Блокирующие:  
[<https://docs.google.com/spreadsheets/d/1uncyVvKie50oKX-Z5LdlOhmu78eGzJS6HP2gaPlkgSY/edit?usp=sharing>]

2. Критичные:  
[<https://docs.google.com/spreadsheets/d/1uncyVvKie50oKX-Z5LdlOhmu78eGzJS6HP2gaPlkgSY/edit?usp=sharing>]

3. Высокий приоритет:  
[<https://docs.google.com/spreadsheets/d/1uncyVvKie50oKX-Z5LdlOhmu78eGzJS6HP2gaPlkgSY/edit?usp=sharing>]

4. Низкий приоритет:  
[<https://docs.google.com/spreadsheets/d/1uncyVvKie50oKX-Z5LdlOhmu78eGzJS6HP2gaPlkgSY/edit?usp=sharing>]

1. Какой баг показался самым критичным?

BR\_3, "Поломка" приложения при принятии заказа, если заказ был отменен пользователем.

1. Такой продукт можно выпускать в релиз? Почему?

Пока не исправлены блокирующие и критические баги, релиз такого продукта должен быть перенесен.

Выводы о проделанной работе

Как для тебя прошла первая практическая часть проекта? Еще раз закрепил знания полученные в ходе всего периода учебы.

С какими сложностями пришлось столкнуться? Неудобно было каждый раз обновлять информацию для Postman и мобильного устройства при смене сервера.

Что получилось хорошо, а что не очень? В целом, доволен, но можно было бы время использовать рациональнее.

Какие мысли остались? Работаем дальше!!!!!