# Планируемый (предполагаемый) вариант проведения мероприятия



# Общие идеи и предложения по проведению мероприятия

Есть идея предложить участникам мероприятия для решения одну или несколько не сильно сложных практических задач, выполнять которые необходимо именно практически, на компьютере, а не на листе бумаге. Как минимум, это интереснее.

Актуален вопрос об уровне сложности задач, так как мы заранее не знаем уровень подготовки возможных участников мероприятия.

Задания будут в основном на базовые знания веб-технологий, умение базово пользоваться какими-либо инструментами (может быть понадобится какой-нибудь сниффер), как он-лайн, так и оффлайн.

Возможно, для ускорения процесса и облегчения задачи участникам хакатона, организаторам нужно будет заранее подготовить рабочее окружение на тех машинах, на которых будут решаться задачи (например, заранее установить и настроить сниффер на работу с браузером)

Но прежде, чем приступать к разбору и решению практических задач, нам нужно понять уровень ИТ-знаний пришедших на мероприятие людей и немного, своими словами рассказать им о тестировании и о профессии тестировщика в целом.

Здесь не должно быть сухой теории тестирования, как, например, цитирование книги Куликова и тому подобное. Нужно рассказать о тестировании своими словами, коротко. Не надо приводить все возможные варианты классификаций тестирования. Надо сделать так, чтобы люди поняли для чего вообще это тестирование придумали и какова цель этой профессии. И, как и говорилось выше, показывать это все нужно на примерах.

После нашего вступительного слова людям, пришедшим на мероприятие будет предложено поиграть в несложный квиз, по итогам которого, возможно, победителям будут вручены призы. Вопросы и варианты ответов в квизе подобраны таким образом, чтобы нам сразу стало понятно – в теме человек или отвечал наугад и в теме не шарит. Дальнейшая беседа будет строиться в зависимости от того насколько правильно люди отвечали на вопросы квиза.

# Варианты практических заданий для использования на мероприятии

С ходу предлагаю несколько задач на знание веб-технологий.

Повторюсь, это просто предложения по задачам. Реально отобранные задачи к мероприятию перечислены в этом документе ниже.

Заметочка: Задачи предполагают развертывание нужных сервисов (поднятие веб-сайта с тестовыми заданиями) в докер-контейнере. Это развертывание должны заранее реализовать организаторы мероприятия.

Учитываем ограниченность времени проведения этого мероприятия и еще то, что мы совершенно не знаем уровень знаний, компетенций, умений и навыков тех, кто притопает на это мероприятие. По-этому вопросы надо формулировать с учетом этих ньюансов. Задача квиза – выявить лучших.

## Задача №1 (никаких БД не используем, чисто HTML + CSS + PHP).

Пользователь заходит на определенный сайт и видит перед собой форму авторизации для ввода логина и пароля с кнопкой входа.

Первый шаг участника – успешно залогиниться на сайт. Валидные для входа значения в чистом виде захардкожены в коде на бэкенде. Валидные креды для входа можно показать участнику в html-коде страницы или в подключенном css в комментариях. Можно показать, как в чистом виде (но это скучно), а можно закодировать во что-то легко раскодируемое какими-нибудь он-лайн инструментами (URL-Encode, Base64, HEX, MD5 и т.д.). Нужно всего лишь найти этот нужный комментарий в коде, расшифровать данные, получить логин и пароль и войти на сайт). После этого юзер попадет на второй шаг задачи.

На втором шаге пользователю нужно ввести валидный 4-х значный пин-код (также захардкожен в исходнике). Подсказку (валидный пин или его часть) пользователю для разнообразия можно отправить в хедере-ответа от сервера, например. А можно в качестве подсказки отправить что-то типа «для получения пин-кода отправь POST-запрос на эндпоинт /pin с телом QAServiceLab» (можно намекнуть на использование CURL).

Прошли оба шага – решили задачу.

Вариант брутфорса тоже возможен, не запрещено. Кто-то возможно сможет воспользоваться враппером и увидеть исходник PHP. Ну это маловероятные варианты развития событий.

## Задача №2 (уже с SQL БД).

Суть будет аналогично задаче №1, но только уже будем использовать SQLite (для простоты). Здесь для решения уже можно будет воспользоваться SQL-инъекциями, например (защиты не будет, WAF тоже не будет). Можно и брутфорсить. Можно также куда-то подсказки спрятать (например показать имена таблиц, что уже сильно облегчит использование SQL-инъекции).

## Задача №3 (например, что-то связанное с JWT-токенами, но это уже посложнее будет).

**ВОТ ЭТУ ЗАДАЧУ МОЖНО ОСТАВИТЬ НА ЗАКУСКУ ДЛЯ САМОСТОЯТЕЛЬНОГО РЕШЕНИЯ УЖЕ ПОСЛЕ ЗАВЕРШЕНИЯ ЭТОГО МЕРОПРИЯТИ. ТО ЕСТЬ СКАЗАТЬ ЛЮДЯМ, ЧТО ЕСТЬ ВОТ ТАКАЯ ВОТ ЗАДАЧА НА ТАКОМ-ТО САЙТЕ. ЗАДАНИЕ БУДЕТ ДОСТУПНО ОГРАНИЧЕННОЕ КОЛИЧЕСТВО ВРЕМЕНИ (НЕДЕЛЮ, НАПРИМЕР). ВСЕ ЖЕЛАЮЩИЕ МОГУТ ЕГО РЕШИТЬ И ПРИСЛАТЬ СВОИ ОТВЕТЫ КУДА-ТО. МОЖНО АНОНСИРОВАТЬ В НАШИХ СОЦ.СЕТЯХ. ЗАДАНИЕ ПОТРЕБУЕТ НЕМНОГО РАЗОБРАТЬСЯ В ВОПРОСЕ САМОСТОЯТЕЛЬНО.**

Задача – залогиниться под админом!

Допустим есть доступ форма авторизации и известны валидные креды какого-то пользователя. Залогинившись под этим пользователем мы получим jwt-токен в куках. Чтобы залогиниться под админом, пользователю придется:

1. Либо подобрать (пробутфорсить, либо мы где-то спрячем этот секретный ключ) secretkey (3-ю часть токена) и создать уже токен для админа и подсунуть его в куки
2. Либо можно сделать наш токен НЕ ИСПОЛЬЗУЯ шифрования. В таком случае 3-я часть токена (секреткей) будет игнорится. Это вариант проще, т.к. пользователю ничего не нужно будит брутфорсить. Можно вообще jwt.io воспользоваться.

## Задача №4 (Самостоятельный поиск багов на импровизированном сайте интернет-магазина кондитерской).

Задача в том, чтобы люди самостоятельно нашли на предложенном сайте специально заложенные туда баги.

Это задание было в BattlePass-2024. Так что список найденных багов у меня есть.

Сайт сделан просто на HTML + CSS + JavaScript. Так что особого окружения разворачивать не придется.

## Задача №5 (Самостоятельный поиск багов в десктопной игре «Змейка»).

Игра «Змейка». Сделана на питоне. Пока не сделана 😊

# Примерный сценарий проведения мероприятия (исключительно моё вИдение происходящего)

**Начало: в 10:30**

*[Все заинтересованные лица уже собрались в классе, где пойдет речь о тестировании. Наше вступительное слово, КВИЗ, рассказ о тестировании с разбором конкретного примера.]*

Здравствуйте! Сегодня мы поговорим о тестировании в целом и профессии тестировщика. Как вы думаете, что такое тестирование ПО и для чего оно вообще нужно?

*[Выслушиваем некоторые варианты ответов и в зависимости от качества и содержания ответов продолжаем, импровизируем.]*

Да, действительно, в чем-то вы правы, а в чем-то не совсем [импровизируем]. Если сказать кратко, то тестирование – это процесс проверки ПО на соответствие фактического поведения и ожидаемого. Прежде, чем продолжить нашу беседу предлагаем вам сыграть в небольшой квиз. Не переживайте, сложных вопросов в квизе нет. По результатам квиза мы сможем понять уровень ваших текущих базовых знаний в сфере тестирования и ИТ-сферы в целом. Победители (кто наберет больше баллов и быстрее всех правильно ответит на вопросы) получат приз.

*[Играем в КВИЗ. Обсуждаем его результаты, награждаем победителей. По результатам квиза сразу поймем кто на что способен. В квизе вопросы и варианты ответов подобраны так, что будет сразу понятно – в теме человек или отвечал наугад и в теме не шарит. Дальнейшая беседа будет строиться в зависимости от того насколько правильно люди отвечали на вопросы квиза.]*

Да вы молодцы (или не молодцы)! Неплохие результаты!

Итак, мы с вами выяснили, что тестирование – это фактически проверка ПО на соответствие фактического поведения и ожидаемого. Собственно говоря, несоответствие фактического и ожидаемого поведения ПО мы и называем багом.

И, если с фактическим поведением все в принципе понятно – это то, как конкретно ведет себя ПО, то, что такое ожидаемое поведение? Как вы думаете?

*[Люди, возможно, что-то там отвечают. В любом случае надо подвести беседу к тому, что ожидаемое поведение берется из документации, требований, из своего личного опыта о том, как логические верно по идее должно вести себя приложение пр.]*

Да, вы правы (или не правы 😊)! Основным источником ожидаемого поведения ПО являются требования к ПО, описанные в документации к продукту. Именно в требованиях документируется то, как программа должна себя вести при наступлении определенных событий.

Конечно, не всегда такие требования существуют (например, если проект очень маленький, или кому-то просто было лень этим заниматься) или существуют, но написаны очень некачественно и неполно. Что же делать в таком случае?

А в таком случае можно и нужно ориентироваться на свой опыт даже не тестирования, а обычного использования ПО, на простую логику о том, как логически корректно должно вести себя приложение. Можно сравнить поведение нашего тестируемого приложения с поведением других похожих приложений в аналогичной ситуации и таким образом собрать для себя какие-то важные требования к его поведению.

Например, при отправке сообщения в каком-то мессенджере вы же ожидаете увидеть на сообщении метку «отправлено» (обычно, одиночную галочку), а не, например, сообщение с текстом «500 Server Error» после которого мессенджер вообще закрывается? Верно же? И вы рассчитываете, что именно так себя и должен вести любой правильно созданный мессенджер 😊

Кстати, одним из этапов тестирования является собственно тестирование требований. Обычно тестировщиков привлекают к этому процессу еще на ранних стадиях разработки ПО, конечно. И чем раньше тестировщики начинают работать над продуктом во время его жизненного цикла – тем лучше, быстрее и финансово дешевле в плане устранения багов и обеспечения качества в целом.

*[Здесь мы кратко скажем, что тестирование бывает функциональным и нефункциональным]*

Как вы уже поняли из квиза или из своего опыта, каждое приложение разработано с какой-то целью и выполняет какие-то конкретные определенные задачи, то есть обладает определенным функционалом. Собственно говоря, тестирование этого функционала приложения и называется функциональным тестированием.

Существует еще и нефункциональное тестирование, которое включает в себя проверку всего остального, что не вошло в функциональное тестирование – нагрузочное тестирование, тестирование безопасности, UI/UX и т.п., но эта тема очень обширная и, к сожалению, у нас просто не хватит времени сегодня поговорить обо всем этом.

А как вы думаете, каким образом и по какой причине появляются баги в ПО?

*[Снова слушаем варианты ответов и дальше уже импровизируем. Наверняка скажут, что причина появления багов – это косяки разработчиков. Если же про это не скажут, то к этому нужно подвести]*

Да, баги в ПО несомненно появляются в результате разработки. Но, это совершенно неправильно винить только разработчиков (программистов) в существовании багов в продукте. Баги могут появиться, например, из-за некорректно составленных требований к программе, то есть еще на этапе только лишь проектирования продукта. При этом разработчики запрограммировали поведение программы строго по составленным требованиям, но это все равно привело к появлению багов. То есть ошибка закралась уже на этапе проектирования требований. А ведь этого можно было избежать или по крайней мере снизить вероятность появления багов.

Этот пример показывает, что начинать обеспечивать качество продукта в идеале нужно на ранних стадиях SDLC продукта и, опять же в идеале, над этим должна работать вся команда, работающая над продуктом (аналитики, разрабы, дизайнеры и т.д.), а не только тестировщики.

*[А тут мы перейдем к тому, что для того, чтобы что-то тестировать – в этом нужно разбираться. И не только в теории тестирования и используемых инструментах, а как минимум нужно обладать какими-то ИТ-знаниями на уровне повыше, чем устанавливать винду.]*

Как вы уже наверняка поняли из квиза (да и не только из квиза) современное ПО довольно сложное и в основном состоит из множества компонентов, каким-то образом связанных друг с другом и взаимодействующих между собой. Кроме того, наше разрабатываемое ПО может использовать в своей работе и стороннее ПО, написанное абсолютно другой командой.

В качестве примера можно привести сервис онлайн обменника валют, который подтягивает курсы валют из какого-то другого источника, совершенно другого приложения, например, с сайта МосБиржи.

Соответственно, чтобы провести действительно качественное тестирование ПО, тестировщик должен обладать достаточно глубокими знаниями о технологиях работы современного ПО и о способах их взаимодействия друг с другом.

Как минимум, понадобятся знания основ сетевых технологий, умения работать с различными базами данных хотя бы на уровне написания базовых запросов и т.д. Естественно, что каждое приложение в чем-то уникально, в том числе и в плане используемых в нем технологий и, естественно, глубоко знать каждую технологию просто невозможно (но можно к этому стремиться). Особенно сейчас, когда ИТ развивается семимильными шагами. По этому хороший тестировщик (и не только тестировщик) должен уметь находить нужную недостающую информацию, попросту – гуглить (пока не заставили исключительно яндексить 😊), чтобы более детально разобраться в интересующем вопросе.

Но! Базовые знания о том, как в общем плане работает технология нужно все-таки иметь.

*[Здесь можно кратко рассказать, что тестирование бывает ручным и автоматизированным. Рассказать, что для тестирования применяется множество различных инструментов, разработанных специально для тестирования. Но, боюсь, что на практическое задание времени не останется. Смотрим уже по месту.]*

Для проведения качественного тестирования придумано и разработано множество различных инструментов, ускоряющих и облегчающих процесс тестирования. Их реально довольно много и сегодня мы их явно не успеем рассмотреть.

*[А тут мы перейдем в DevTools и разбору нашего примера – задания с сайтом авторизации.]*

Но, чтобы наша с вами сегодняшняя беседа не превратилась в обычную скучную лекцию, есть предложение вместе с вами разобрать одну практическую и, надеюсь, интересную задачку. Для решения этой задачи вам понадобятся базовые знания HTML (кстати, а что это такое?), внимательность, немного логики и совсем чуть-чуть математики.

В процессе разбора решения это задачи мы увидим некоторые возможности языка HTML и CSS, а также то, к чему может привести, например, забывчивость и невнимательность разработчиков при создании приложения.

*[Открываем сайт с задачей]*

Как видим, перед нами открылся сайт со страницей авторизации. Для того, чтобы решить задачу нужно всего лишь авторизоваться на сайте, введя в форму авторизации корректный логин и пароль.

Метод брутфорса, то есть подбора логина и пароля путем перебора мы использовать не будем. Да, этот способ возможен, но, поверьте, в данном случае перебор всех комбинаций займет очень-очень много времени. Можете даже не пытаться.

Для решения этой задачи достаточно использовать инструмент, который встроен практически в любой современный браузер.

Вы догадались о каком инструменте идет речь?

*[Речь идет о DevTools. Люди что-то отвечают – может догадаются, а может и нет. Не важно.]*

Итак, практически в каждом популярном браузере этот инструмент имеется в наличии и называется он «Инструменты разработчика» или Development Tools, сокращенно – DevTools.

*[Говорим, как его – DevTools – вызвать.]*

Предлагаю, вам попробовать начать решать задачу самостоятельно, а в случае необходимости, мы будет давать вам подсказки и обсуждать ваши предложения. Внимательно исследуйте страницу и попытайтесь все-таки войти на сайт, найдя здесь правильные подсказки и правильные учетные данный. Поверьте, они здесь точно есть!

*[Тут уж я не знаю, что писать, так как прогнозировать, как пойдет решение и обсуждение задачи сейчас невозможно. Импровизируем.]*

*[После завершения разбора этой задачи предлагаем на полностью самостоятельное решение еще одну или две задачи (можно на выбор). Цель задания – самостоятельно найти и своими словами описать найденные баги в приложении.]*

# Отобранные практические задачи для мероприятия

## Задача №1 (для совместного решения и разбора)

**Цель задания:** Есть небольшой сайт. На главной странице сайта есть форма авторизации. Цель задания – успешно залогиниться на сайте, а потом еще и ввести корректный пин-код.

Для достижения цели и успешного решения задачи нужно найти на этом сайте спрятанные здесь же учетные данные. Повторюсь, все нужные данные каким-то образом спрятаны непосредственно на самом сайте.

Подробнее о задании и о том, где и что здесь спрятано:

- Все логины и пароли захардкожены в исходнике на бэкенде.

- Под формой авторизации, справа от надписи "Не знаю, что делать..." имеется невидимая кнопка (точнее, ссылка), нажимая на которую пользователю будут показываться подсказки о том в какую сторону ему двигаться для решения задачи.

- Форма авторизации имеет фоновое изображение, на котором в открытом виде изображен один из валидных кредов (и логин и пароль). При этом фоновое изображение по умолчанию перекрыто сплошным цветом. Если в DevTools отключить эту заливку сплошным цветом, то увидим фоновое изображение с нужными нам данными.

- В форме авторизации поле ввода пароля ограничено до 10 символов. При этом все захардкоженые пароли имеют длину более 10 символов. Чтобы ввести корректный пароль нужно это дело исправить. Подсказка об этом имеется в коде страницы в виде комментария.

- Также в HTML-коде страницы в виде комментария в чистом виде спрятан еще один из вариантов валидного логина. А пароль для этого валидного логина спрятан в виде комментария в подключенном CSS-файле. Для усложнения задания пароль закодирован в Base64.

- После ввода валидных кредов (логина и пароля) в форме авторизации пользователю будет нужно ввести корректный пин-код (откроется уже другая форма), который также захардкожен в исходнике бэкенда (правильный пин-код: 1973). Заметим, что фон страницы обновился. На этом фоне есть явный намек на то, что для решения задачи, то есть нахождения правильного пин-кода, нужно произвести перевод некоего числа из двоичной системы исчисления в десятичную. Сам пин-код в двоичной системе исчисления также содержится на этом фоновом изображении.

- Задача считается выполненной после ввода всех валидных кредов - логина, пароля и пин-кода.

## Задача №2 (уже для самостоятельного решения)

**Цель задания:** Задача состоит в том, чтобы люди самостоятельно нашли на предложенном сайте специально заложенные туда баги. Сайт – лайтовый вариант интернет-магазина кондитерской.

То есть нужно провести исследовательское тестирование сайта, найти и описать найденные там баги.

Это задание было в BattlePass-2024. Так что список найденных багов у меня есть. Сюда я их вставлять не буду – они перечислены в отдельном документе.

Сайт сделан просто на HTML + CSS + JavaScript. Так что особого окружения разворачивать не придется. Задание не мое, а честно изъятое из открытых источников 😊

## Задача №3 (тоже для самостоятельного решения)

**Цель задания:** Есть десктопная игра «Змейка» под Windows (EXE-шник под Windows). Задача также состоит в том, чтобы найти и описать специально заложенные туда баги.

Написана на питоне. Точнее, пока еще не написана 😊

Задача состоит в том, чтобы люди самостоятельно нашли на предложенном сайте специально заложенные туда баги. Сайт – лайтовый вариант интернет-магазина кондитерской.

То есть нужно провести исследовательское тестирование сайта, найти и описать найденные там баги.

Это задание было в BattlePass-2024. Так что список найденных багов у меня есть. Сюда я их вставлять не буду – они перечислены в отдельном документе.

Сайт сделан просто на HTML + CSS + JavaScript. Так что особого окружения разворачивать не придется. Задание не мое, а честно изъятое из открытых источников 😊

- Тестирование сайта

- Тестирование игры «Змейка»