Частное учреждение профессионального образования

«Высшая школа предпринимательства (колледж)»

(ЧУПО «ВШП»)

Кафедра информационных технологий

Внедрение и поддержка компьютерных систем

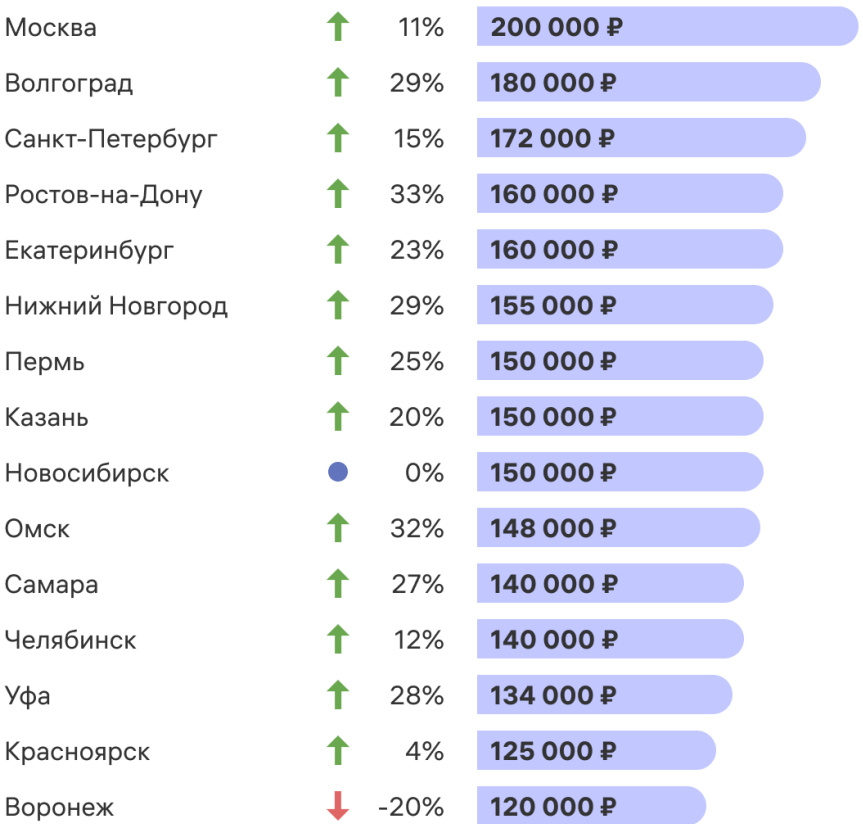
Для разрядки и настройки

В Минцифры заявили о росте зарплат айтишников почти на 20%

Замглавы Минцифры Паршин: зарплаты в IT-сфере выросли на 19%

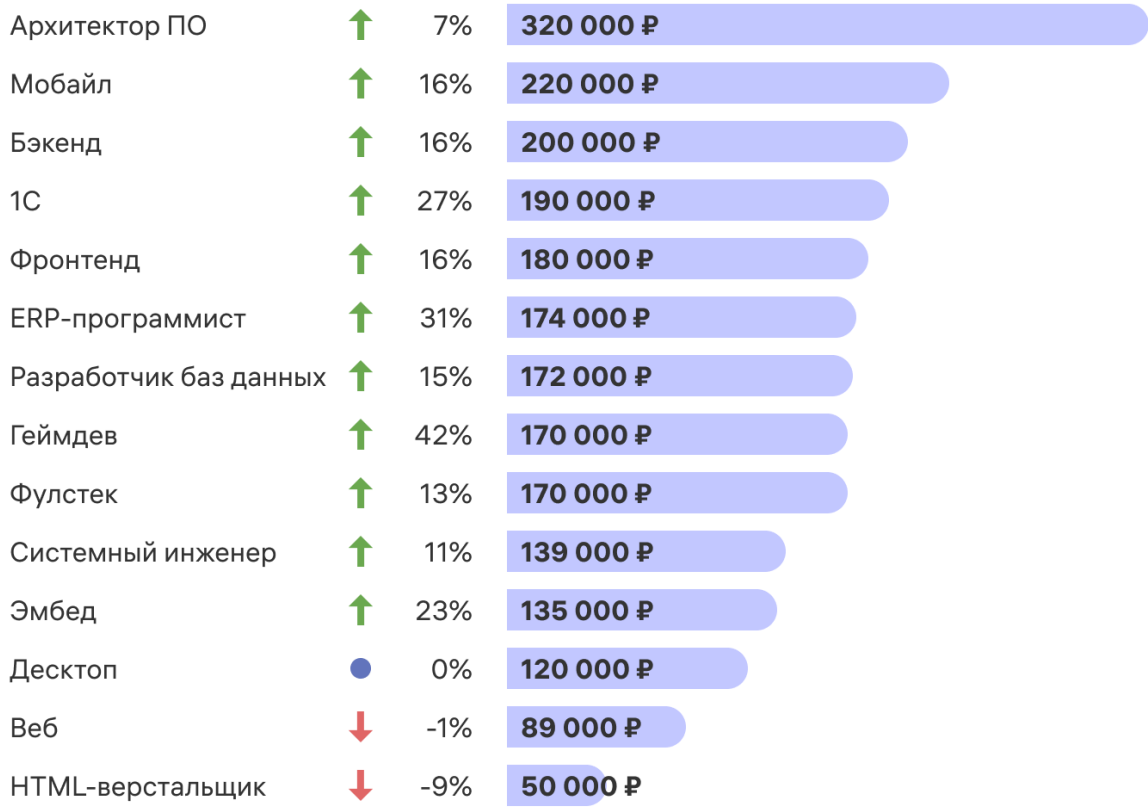
Динамика зарплат айтишников по городам

% — разница между первым полугодием 2023 и вторым полугодием 2022



Динамика зарплат разработчиков по специализациям

% — разница между первым полугодием 2023 и вторым полугодием 2022



Профессия	Россия в целом	СЗФО	Москва	Санкт-Петербург
BI-аналитик, аналитик данных	13,2	67	9,3	18,3
DevOps-инженер	3,9	26,3	2,1	4,2
Аналитик	9,8	18	8,8	10,5
Гейм-дизайнер	46,3	20	25,8	26,9
Дата-сайентист	22,3	19	14,4	31,3
Дизайнер, художник	19,1	46,9	14,5	22
Маркетолог-аналитик	9,6	14,3	8,9	11,1
Интернет-маркетолог	7,2	9,8	7,8	7,9
Менеджер продукта	13,2	22,4	11,8	15,6
Программист, разработчик	5,1	10,2	3,6	5,2
Продуктовый аналитик	19,3	33,5	15	23,1
Руководитель группы разработки	10,8	33,1	10,3	8,3
Руководитель отдела аналитики	26,7	60,5	26,5	14,3
Системный администратор	8,4	11,3	6,7	8,7
Системный аналитик	5,3	22,3	3,7	4,9
Специалист по информационной безопасности	5,8	6,2	3,9	4,9
Тестировщик	14,7	72,7	7,4	15,2


ВЖЖУХ

Принципы тестирования

Принцип 1 — Тестирование демонстрирует наличие дефектов
Принцип 2 — Исчерпывающее тестирование невозможно
Принцип 3 — Раннее тестирование
Принцип 4 — Скопление дефектов (принцип Парето)
Принцип 5 — Парадокс пестицида
Принцип 6 — Тестирование зависит от контекста
Принцип 7 — Заблуждение об отсутствии ошибок

Верификация и валидация

Верификация

Верификация всегда отвечает на вопрос **"делаем ли мы продукт правильно?"**.
Эта проверка связана с тем, что мы убеждаемся в том, что система хорошо спроектирована и безошибочно.

Валидация

Это динамический процесс, в отличии от **верификации**. Этот процесс всегда включает в себя запуск кода программы и отвечает на вопрос **"делаем ли мы правильный продукт?"**

Атрибуты отчета о дефекте

Уникальный идентификатор (ID)
Тема (краткое описание, Summary)
Окружение (Environment)
Серьёзность дефекта (важность, Severity)
Приоритет дефекта (срочность, Priority)
Шаги для воспроизведения (Steps To Reproduce)
Ожидаемый результат (Expected result)
Фактический результат (Actual result)

Severity vs Priority

Severity vs Priority

Серьёзность (severity) показывает степень ущерба, который наносится проекту существованием дефекта. Severity выставляется тестировщиком.

Срочность (priority) показывает, как быстро дефект должен быть устранён. Priority выставляется менеджером, тимлидом или заказчиком

Тестовые артефакты

Чек-лист

Тест-кейс

Use-cases (юз-кейсы)

Use-cases (юз-кейс) — это техника в управлении проектами и разработке ПО, которая используется для описания функциональных требований к системе или приложению.

Он описывает, как пользователь будет использовать систему, чтобы выполнить конкретную задачу. Расписывает но не так подробно — какие шаги он должен предпринять и как система должна реагировать на каждый из этих шагов.

Use-cases являются одним из артефактов тестовой документации.

Основные атрибуты use-cases

1. **Название юз-кейса:** краткое описание того, что делает пользователь.
2. **Акторы:** список всех пользователей, которые могут использовать систему или приложение.
3. **Описание:** детальное описание, как пользователь будет использовать систему или приложение для выполнения конкретной задачи.
4. **Предусловия:** условия, которые должны быть выполнены до начала использования системы.
5. **Шаги использования:** пошаговое описание того, что пользователь должен сделать, чтобы выполнить задачу.
6. **Альтернативные шаги:** шаги, которые могут быть выполнены, если основные шаги не сработают.
7. **Результаты:** описание того, что должно произойти после выполнения задачи.
8. **Ограничения:** описание ограничений или ограничений, которые могут влиять на использование системы.

Простой пример use-cases

Покупка товара в интернет-магазине
Название: Покупка товара
Акторы: Пользователь, Интернет-магазин.
Описание: Пользователь заходит на сайт интернет-магазина, выбирает товары, которые он хочет купить, добавляет их в корзину и проходит процесс оформления заказа.
Предусловия: Пользователь должен быть зарегистрирован на сайте.
Шаги использования:

1. Пользователь выбирает товары, которые он хочет купить, и добавляет их в корзину.
2. Пользователь переходит на страницу оформления заказа.
3. Пользователь заполняет форму с личными данными и адресом доставки.
4. Пользователь выбирает способ оплаты и подтверждает заказ. Результаты: Заказ успешно оформлен, и пользователь получает подтверждение на свой электронный адрес.

Главные отличия от тест-кейса

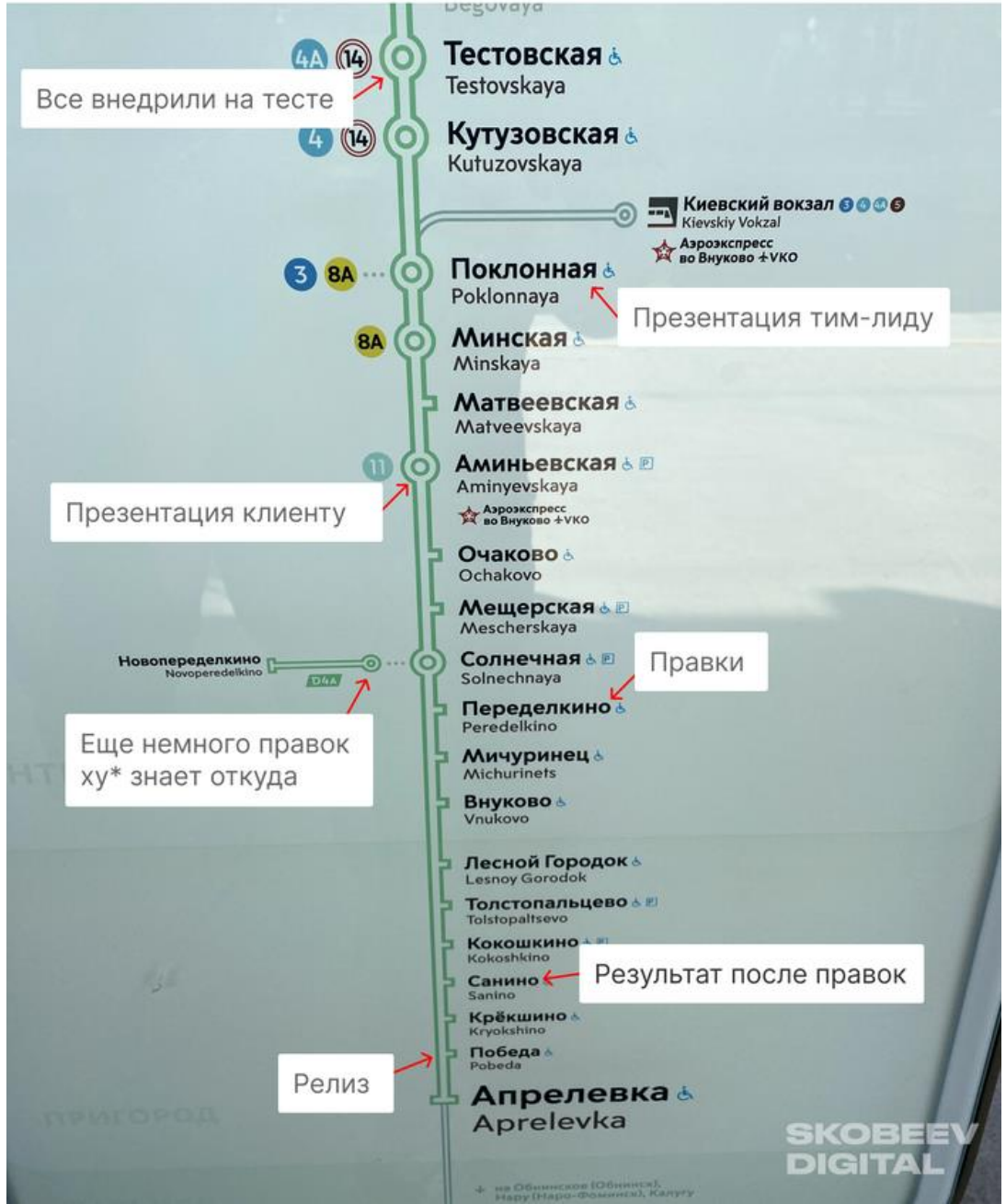
Основное отличие между **use-cases** и **тест-кейсом** заключается в том, что use-cases описывает более широкий контекст использования системы.
Тогда как **тест-кейс** является более узким вариантом, который описывает только тестирование конкретной функции или возможности системы.
Use-cases может включать **множество тест-кейсов** внутри себя, а также другие элементы, такие как пользовательский интерфейс, отчетность и взаимодействие с другими системами.

Бизнес-аналитики могут использовать use-cases диаграммы для определения требований и предоставления разработчикам более детальной информации о том, как система должна работать.
Разработчики могут использовать use-case диаграммы в качестве основы для разработки, а также для тестирования и отладки своего кода.
Тестировщики могут использовать use-case диаграммы для определения тест-кейсов и проверки соответствия требованиям.
Менеджеры проектов могут использовать их для контроля за процессом разработки и оценки прогресса.

Важные моменты, которые следует учитывать при создании use-cases

1. **Определение акторов:** Необходимо ясно определить, кто является основными участниками процесса, который описывается в use-cases. Это могут быть пользователи, клиенты, поставщики или любые другие люди или системы, взаимодействующие в рамках определенной функциональности.
2. **Описание шагов.** Важно описать каждый шаг процесса в деталях, начиная с начального состояния, процесса выполнения действий и заканчивая конечным состоянием.
3. **Необходимо определить предусловия и постусловия.** Предусловия — это условия, которые должны быть выполнены перед началом процесса, а постусловия — это результаты, которые должны быть достигнуты по завершении процесса. Оба этих аспекта важны для правильного понимания процесса.
4. **Включение альтернативных сценариев.** Use-case не всегда будет выполняться в идеальных условиях, поэтому необходимо включать альтернативные сценарии, которые могут возникнуть в процессе выполнения.
5. **Определение завершения процесса.** Важно ясно определить, как процесс завершается. Это может быть достижение определенной цели, получение определенного результата или ошибка, которая приводит к неудачному завершению процесса.

Roadmap твоего проекта



Вид тестирования

Статическое тестирование
Динамическое тестирование

- **Ручное тестирование**
- **Автоматизированное тестирование**

- **Позитивное тестирование**
- **Негативное тестирование**

Нефункциональное тестирование

Стрессовое тестирование (stress testing)
Тестирование удобства использования (usability testing)
Тестирование локализации (localization testing)
Тестирование безопасности (security testing)

Тест-дизайн

Тест-дизайн

Тест-дизайн(Test design) - это этап процесса тестирования нашего программного обеспечения, на котором проектируются и создаются **тест-кейсы**, в соответствии с определенными ранее критериями качества и целями тестирования.

Техники тест-дизайна

Тестирование на основе классов эквивалентности (equivalence partitioning)

Техника анализа граничных значений (boundary value testing)

Тестирование на основе классов эквивалентности

Техника, при которой мы разделяем функционал (часто диапазон возможных вводимых значений) на группы эквивалентных по своему влиянию на систему значений.

Такое разделение помогает убедиться в правильном функционировании целой системы — одного класса эквивалентности, проверив только один элемент этой группы.

Эта техника заключается в разбиении всего набора тестов на классы эквивалентности с последующим сокращением числа тестов.

Признаки эквивалентности тестов

- направлены на поиск одной и той же ошибки;
- если один из тестов обнаруживает ошибку, другие скорее всего, тоже её обнаружат;
- если один из тестов не обнаруживает ошибку, другие, скорее всего, тоже её не обнаружат;
- тесты используют схожие наборы входных данных;
- для выполнения тестов мы совершаем одни и те же операции;
- тесты генерируют одинаковые выходные данные или приводят приложение в одно и то же состояние;
- все тесты приводят к срабатыванию одного и того же блока обработки ошибок;
- ни один из тестов не приводит к срабатыванию блока обработки ошибок.

Классический пример

Есть поле ввода с диапаз ачений от 1 до 100.

На 95 тестов на допустимые значения и на несметное количество тестов на недопустимые значения уйдет очень много времени. И здесь нам помогут классы эквивалентности.
Все допустимые значения могут влиять на поле ввода одинаково, следовательно **все числа от 1 до 100 можно смело считать эквивалентными**.
С другой стороны, все недопустимые значения должны одинаково влиять на поле ввода (в идеале не должно быть возможности ввода этих значений в поле).

Классический пример

Таким образом, есть уже несколько классов эквивалентности:

- **Допустимые значения (от 1 до 100);**
- **Недопустимые значения:**
  1. **от — ∞ до 0;**
  2. **от 101 до + ∞;**
  3. **специальные символы (# @ + — / _ : ; “ ‘ и т.д.);**
  4. **буквы.**

Используя классы эквивалентности можно протестировать поле ввода минимум из **5 тестов**.

На практике классы эквивалентности обязательны при тестировании всевозможных форм и полей ввода.

Плюсы и минусы техники анализа эквивалентных классов

К плюсам можно отнести **отсеивание огромного количества значений ввода**, использование которых просто бессмысленно.

- **Эффективность:** позволяет значительно сократить количество тестов, необходимых для проверки функциональности, за счет того, что тесты группируются в классы.
- **Простота:** техника является простой в понимании и применении, даже для новичков в тестировании.
- **Улучшение качества тестирования:** позволяет более полно покрыть функциональность, так как тесты охватывают все возможные сценарии использования.

К минусам можно отнести **неправильное использование техники**, из-за которого есть риск упустить баги.

- **Невозможность учесть все возможные входные данные:** техника предполагает, что все возможные входные данные можно разбить на классы, что может быть невозможно в случае большого количества входных параметров.
- **Ограниченность использования:** техника наиболее эффективна для функций с дискретным набором входных параметров.
- **Необходимость тщательного анализа:** чтобы использовать технику, необходимо провести тщательный анализ всех возможных входных параметров , что может быть трудоемким процессом.


ОТ АЙТИ НЕ УБЕЖИШЬ!
У НАС ВЕЧНЫЙ СПРИНТ

Техника анализа граничных значений

Граничные значения — это те места, в которых один класс эквивалентности переходит в другой.

Граничные значения — это значения на границе допустимого диапазона входных данных, которые могут привести к изменению поведения программы.
Это могут быть значения, которые являются минимальными или максимальными для определенного типа данных, значения, близкие к ним, или значения, которые приводят к переполнению буфера или другим ошибкам. Использование граничных значений в тестировании помогает выявлять ошибки, связанные с обработкой граничных условий. Например, если программа обрабатывает числа в диапазоне от 1 до 100, то граничные значения будут 1 и 100. Тестирование с использованием этих значений позволит выявить ошибки, связанные с обработкой крайних значений.

Техника анализа граничных значений

На каждой границе диапазона следует проверить по три значения:

- **граничное значение;**
- **значение перед границей;**
- **значение после границы.**

Цель этой техники — найти ошибки, связанные с **граничными значениями**.

Алгоритм использования техники граничных значений

- **выделить классы эквивалентности;**
- **определить граничные значения этих классов;**
- **нужно провести тесты по проверке значения до границы, на границе и сразу после границы.**

Количество тестов для проверки граничных значений будет равен количеству границ, умноженному на 3.
Рекомендуется проверять значения **вплотную к границе**.
К примеру, есть диапазон целых чисел, граница находится в числе 100.
Таким образом, будем проводить тесты с числом 99 (до границы), 100 (сама граница), 101 (после границы).

Таблица принятия решений

Способ компактного представления модели со сложной логикой; инструмент для упорядочения сложных бизнес требований, которые должны быть реализованы в продукте. Это **взаимосвязь между множеством условий и действий**.

Эта техника основывается на принципе, что каждый тест-кейс должен проверять конкретный функциональный аспект приложения. Для достижения этой цели используется таблица принятия решений, которая позволяет разработчикам определить, какие варианты использования приложения следует проверять.

Таблица принятия решений

Таблица принятия решений представляет собой таблицу с двумя осями — вертикальной и горизонтальной. В вертикальной оси перечислены функциональные аспекты приложения, которые должны быть проверены, в то время как в горизонтальной оси перечислены различные варианты использования приложения.

Каждый элемент таблицы содержит информацию о том, следует ли проверять соответствующий функциональный аспект для данного варианта использования приложения.

Пример таблицы принятия решений для приложения для заказа еды выглядит следующим образом:

ФУНКЦИОНАЛЬНЫЙ АСПЕКТ	ВАРИАНТ ИСПОЛЬЗОВАНИЯ	ПРОВЕРКА
Регистрация нового пользователя	Регистрация через электронную почту	Да
Регистрация нового пользователя	Регистрация через социальные сети	Да
Авторизация пользователя	Авторизация через электронную почту	Да
Авторизация пользователя	Авторизация через социальные сети	Да
Просмотр меню ресторанов	Просмотр меню ресторана	Да
Просмотр меню ресторанов	Поиск ресторана по названию	Да
Добавление товаров в корзину	Добавление товара в корзину	Да
Добавление товаров в корзину	Изменение количества товаров в корзине	Да
Оформление заказа	Выбор способа доставки	Да
Оформление заказа	Выбор способа оплаты	Да

Таблица принятия решений

Таблица принятия решений, как правило, разделяется на 4 квадранта

Условия	Варианты выполнения действий
Действия	Необходимость действий

- ***Условия*** — список возможных условий.
- ***Варианты выполнения действий*** — комбинация из выполнения и/или невыполнения условий этого списка.
- ***Действия*** — список возможных действий.
- ***Необходимость действий*** — указание надо или не надо выполнять соответствующее действие для каждой из комбинаций условий.

Таблица принятия решений — плюсы и минусы

ПЛЮСЫ ИСПОЛЬЗОВАНИЯ ТАБЛИЦЫ ПРИНЯТИЯ РЕШЕНИЙ	МИНУСЫ ИСПОЛЬЗОВАНИЯ ТАБЛИЦЫ ПРИНЯТИЯ РЕШЕНИЙ
Позволяет систематизировать проблему и найти наиболее эффективное решение	Может быть трудно определить критерии оценки
Упрощает принятие решений на основе объективных критериев	Важность критериев может быть недостаточно точно определена
Позволяет сравнивать различные альтернативы	Некоторые альтернативы могут быть недостаточно изучены
Помогает избежать влияния эмоций и предубеждений на принятие решений	Не учитывает некоторые факторы, которые могут оказывать влияние на принятие решений
Позволяет быстро и эффективно принимать решения	Может привести к упрощению сложной проблемы
Позволяет объективно оценить риски каждой альтернативы	Может быть сложно учесть все возможные альтернативы и критерии

Методы тестирования

Метод
черного ящика

Метод
серого ящика

Метод
белого ящика

Тестирование чёрного ящика

Тестирование чёрного ящика — также известное как тестирование, основанное на спецификации или тестирование поведения — техника тестирования, основанная на работе исключительно с внешними интерфейсами тестируемой системы.

Согласно **ISTQB**, тестирование черного ящика — это:

- тестирование, как функциональное, так и нефункциональное, не предполагающее знания внутреннего устройства компонента или системы;
- тест-дизайн, основанный на технике черного ящика — процедура написания или выбора тест-кейсов на основе анализа функциональной или нефункциональной спецификации компонента или системы без знания ее внутреннего устройства.

Тестирование белого ящика

Тестирование белого ящика — метод тестирования ПО, который предполагает, что внутренняя структура/устройство/реализация системы известны тестировщику.

Согласно **ISTQB**, тестирование белого ящика — это:

- тестирование, основанное на анализе внутренней структуры компонента или системы;
- тест-дизайн, основанный на технике белого ящика — процедура написания или выбора тест-кейсов на основе анализа внутреннего устройства системы или компонента.

Почему **«белый ящик»**? Тестируемая программа для тестировщика — прозрачный ящик, содержимое которого он прекрасно видит.

Тестирование серого ящика

Тестирование серого ящика — метод тестирования ПО, который предполагает комбинацию White Box и Black Box подходов.
То есть, внутреннее устройство программы нам известно лишь частично.

Что такое ISTQB – International Software Qualifications Board?

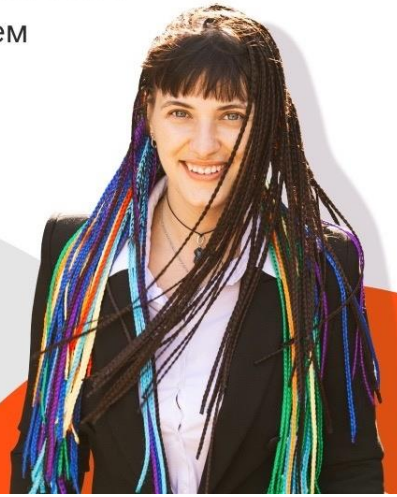
ISTQB - некоммерческая организация, занимающаяся определением различных принципов развития сферы тестирования ПО, таких как структура и правила аккредитации, сертификации и т.п.
Рабочие группы **ISTQB** отвечают за разработку и поддержку сертификационных программ и экзаменов. **ISTQB** включает в себя Национальных и Региональных представителей.

ISTQB организовывает сертификацию тестировщиков по всему миру. Сейчас доступно два уровня сертификации: Базовый уровень (Foundation Level) – один модуль, и Продвинутый уровень (Advanced Level) – три модуля.
Один из основных принципов сертификации состоит в строгом разделении администрации, проводящей экзамены от тех, кто готовит к экзаменам по соответствующим **ISTQB-курсам**.

Стажировка РТК

Лучший способ усилить свою команду тем, кого можно назвать гордыми буквами QA, – вырастить этого специалиста самостоятельно

В Школе тестировщика «Ростелекома» преподают коллеги из разных направлений и проектов, их всех объединяет желание делиться знаниями и помогать расти тем, кто, как и мы, хочет сделать этот мир лучше. Даже если для этого придется сломать пару систем и положить на лопатки тестовый стенд ;)



Ася Радужная
директор департамента тестирования и обеспечения качества

Что ждет наших студентов?

Начнем с классической теории: принципов, процесса, этапов, жизненных циклов тестирования и анализа ТЗ

Разберем тестовую документацию и научим ее составлять

И, конечно, углубимся в практику тест-дизайна, тестирования API и web, а также применим SQL

Я верю, что первый поток нашей Школы тестировщика не только поможет начинающим тестировщикам обрести новые знания, но и подарит нам новых коллег, которые останутся в компании после стажировки, будут расти и развиваться вместе с «Ростелекомом»

vk.com/wall-144207470_10817

Наследование

Наследование – это переход имущества умершего гражданина (наследодателя) к другим лицам, то есть наследникам.

Инкапсуляция

Инкапсуляция – это упаковка данных и функций в один компонент (например, класс) и последующий контроль доступа к этому компоненту, создавая тем самым "чёрный ящик" из объекта.

Полиморфизм

Полиморфизм – явление, при котором вещество одного химического состава образует разные кристалические.

ДЗ

1.Форкнуть: **github.com/imerofeev/MDK_04.01_HW**

2.Внимательно прочитать задание!!!1

3.Выполнить и **прислать пуллреквест** (минимум **2** задания, для зачета выполнения)

дедлайн: 26-10-2023 23:59:59

Учебная программа от EPAM по фронтенду

rs.school/js/

старт: 05 ноября