Частное учреждение профессионального образования

«Высшая школа предпринимательства (колледж)»

(ЧУПО «ВШП»)

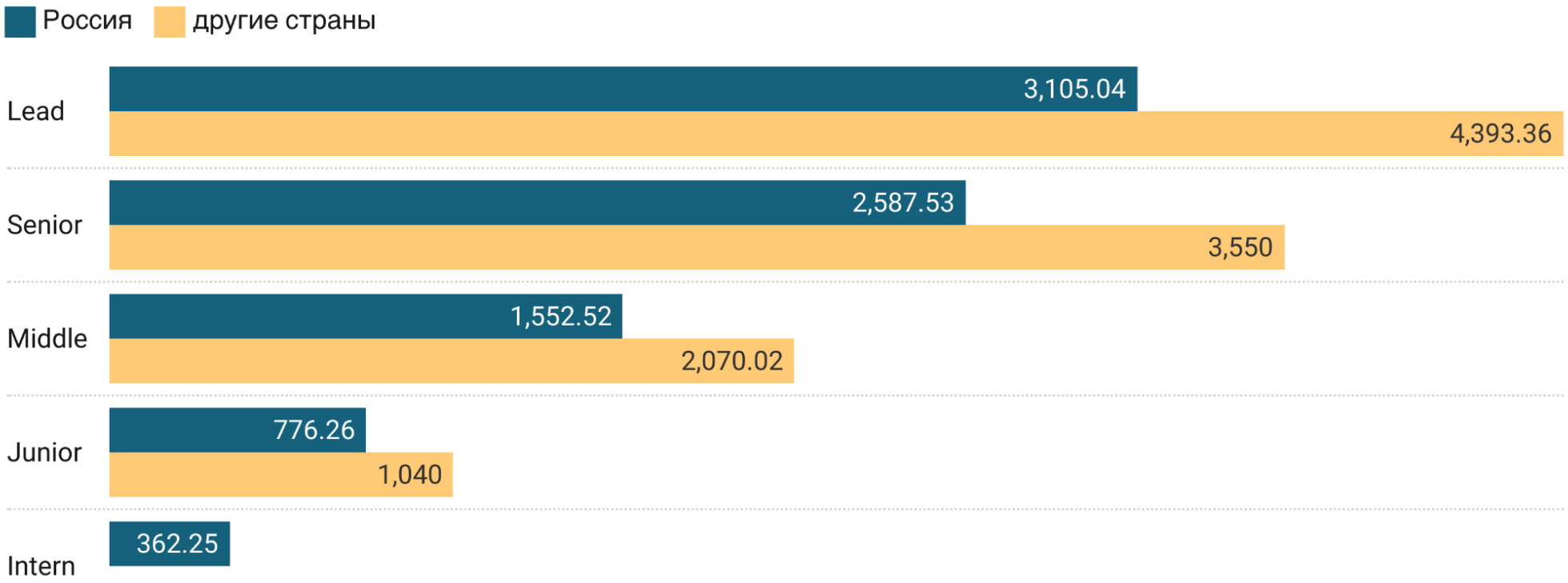
Кафедра информационных технологий

Внедрение и поддержка компьютерных систем

Для разрядки и настройки

Медианная зарплата

Распределение по уровню

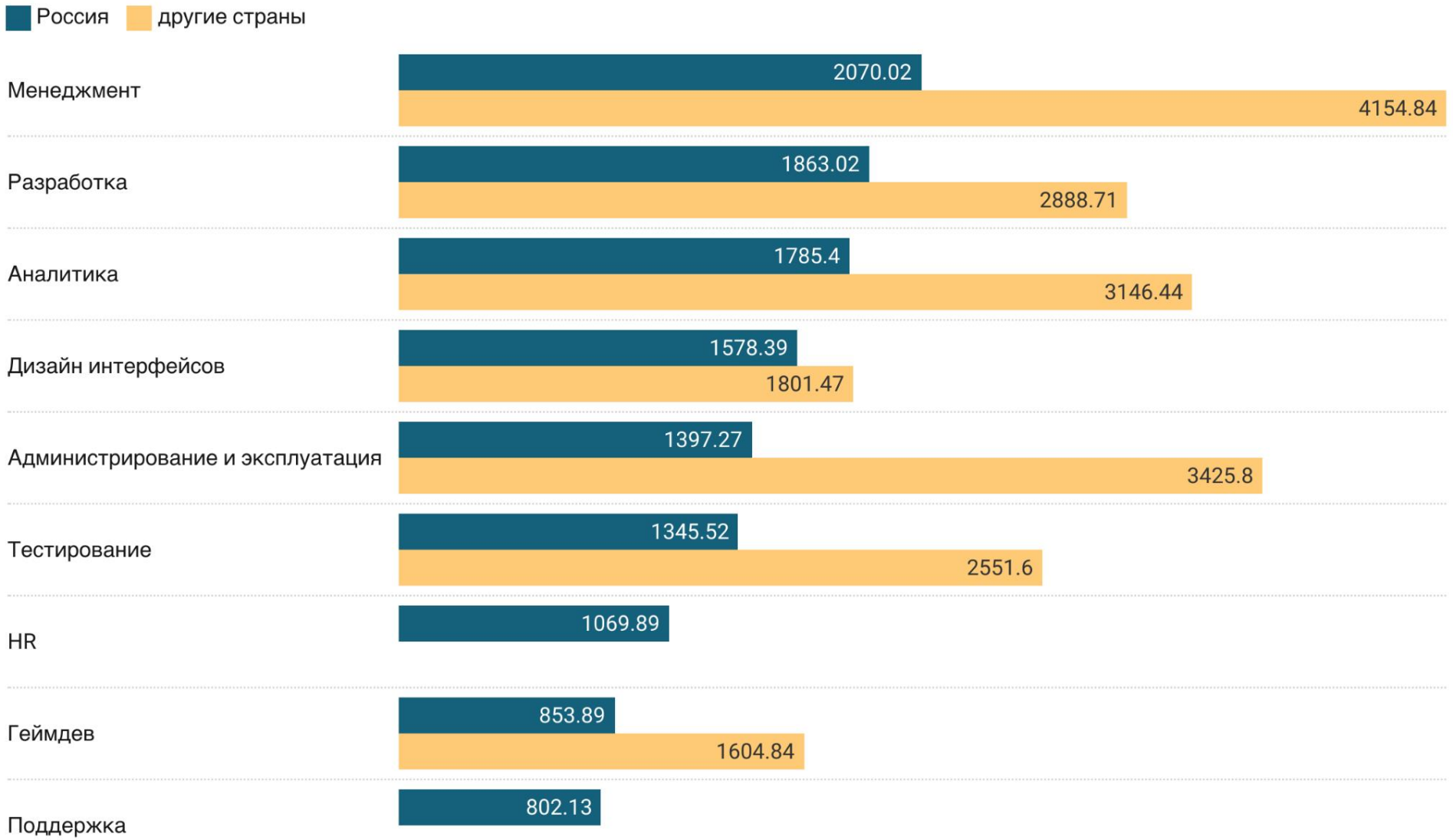


$ в месяц, по курсу валют ЦБ РФ на 21.09.2023

Source: t.me/ruitunion • Created with Datawrapper

Медианная зарплата

Распределение по специализации



$ в месяц, по курсу валют ЦБ РФ на 21.09.2023

Source: t.me/ruitunion • Created with Datawrapper

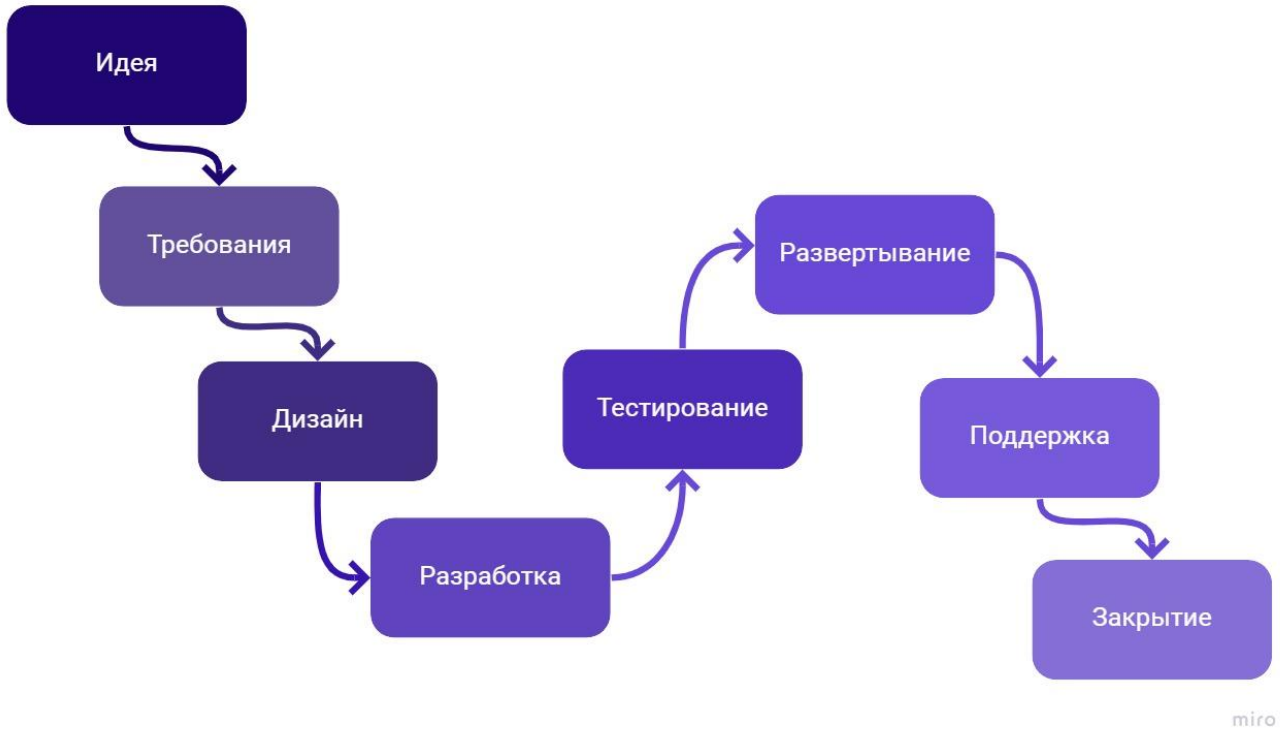
В США инженер проверял конфигурацию кластера серверов медцентра, а его кот прыгнул на клавиатуру и удалил все настройки

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

Этапы жизненного цикла разработки

Идея
Определение требований
Дизайн
  (архитектура) системы
Разработка
Тестирование
Развертывание
Поддержка
Закрытие



Testing

Обеспечение качества (QA — Quality Assurance)

Контроль качества (QC — Quality Control)

Верификация и валидация

Верификация

Это статическая практика проверки документов, дизайна, архитектуры, кода и так далее.
Процесс **верификации** проходит без запуска кода.
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.
Верификация всегда происходит *ДО* **валидации**.

Валидация

Процесс оценки конечного продукта, когда необходимо проверить: соответствует ли программное обеспечение ожиданиям и требованиям клиента.
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?"**.
Процесс **валидации** всегда происходит *ПОСЛЕ* **верификации**, поэтому на этапе **валидаци**и можно уловить ошибки, которые процесс **верификации** не позволил нам найти.

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

Градация Серьезности дефекта (Severity)

Блокирующий (S1 – Blocker)
Критический (S2 – Critical)
Значительный (S3 – Major)
Незначительный (S4 – Minor)
Тривиальный (S5 – Trivial)

Градация Приоритета дефекта (Priority):

P1 Высокий (High)
P2 Средний (Medium)
P3 Низкий (Low)

Схема выполнения рабочей задачи

Тестовые артефакты

Чек-лист

Некий список проверок, в котором показано, что мы будем тестировать и, как следствие, результат и статус данных проверок.

Сам этот список содержит в себе то, что мы собрались вообще сделать, что мы не хотим забыть и что непосредственно мы будем проверять.

Тестовые артефакты

Тест-кейс
Данный вид тестовой документации отличается следующим: во-первых, здесь у нас уже будут шаги, в отличие от чек-листа, в котором мы говорили **ЧТО** мы будем тестировать, здесь мы уже расписываем **КАК** мы будем тестировать.
Опять же, здесь будет результат наших проверок. Также в тест-кейсе обязательно

должен быть заголовок. Возможно будет приоритет, либо же порядок выполнения

нашего тест кейса и еще ряд некоторых атрибутов.

Самое главное, что вы должны усвоить при составлении тест-кейса, что здесь у вас есть некоторая троица основных атрибутов: **пошаговый сценарий(шаги)**, **ожидаемый результат** и **фактический результат**.

Вид тестирования

Классификация по запуску кода на исполнение

Статическое тестирование — процесс тестирования, который проводится для верификации практически любого артефакта разработки: программного кода компонент, требований, системных спецификаций, функциональных спецификаций, документов проектирования и архитектуры программных систем и их компонентов.

Динамическое тестирование — тестирование проводится на работающей системе, не может быть осуществлено без запуска программного кода приложения.

Классификация по доступу к коду и архитектуре

Тестирование белого ящика — метод тестирования ПО, который предполагает полный доступ к коду проекта.

Тестирование серого ящика — метод тестирования ПО, который предполагает частичный доступ к коду проекта (комбинация White Box и Black Box методов).

Тестирование чёрного ящика — метод тестирования ПО, который не предполагает доступа (полного или частичного) к системе. Основывается на работе исключительно с внешним интерфейсом тестируемой системы.

Классификация по уровню детализации приложения

Модульное тестирование — проводится для тестирования какого-либо одного логически выделенного и изолированного элемента (модуля) системы в коде. Проводится самими разработчиками, так как предполагает полный доступ к коду.
Интеграционное тестирование — тестирование, направленное на проверку корректности взаимодействия нескольких модулей, объединенных в единое целое.
Системное тестирование — процесс тестирования системы, на котором проводится не только функциональное тестирование, но и оценка характеристик качества системы — ее устойчивости, надежности, безопасности и производительности.
Приёмочное тестирование — проверяет соответствие системы потребностям, требованиям и бизнес-процессам пользователя.

Классификация по степени автоматизации

Ручное тестирование

Автоматизированное тестирование

Классификация по принципам работы с приложением

Позитивное тестирование — тестирование, при котором используются только корректные данные.

Негативное тестирование — тестирование приложения, при котором используются некорректные данные и выполняются некорректные операции.

Классификация по уровню функционального тестирования

Дымовое тестирование (smoke test) — тестирование, выполняемое на новой сборке, с целью подтверждения того, что программное обеспечение стартует и выполняет основные для бизнеса функции.

Тестирование критического пути (critical path) — направлено для проверки функциональности, используемой обычными пользователями во время их повседневной деятельности.

Расширенное тестирование (extended) — направлено на исследование всей заявленной в требованиях функциональности.

Классификация в зависимости от исполнителей

Альфа-тестирование — является ранней версией программного продукта. Может выполняться внутри организации-разработчика с возможным частичным привлечением конечных пользователей.

Бета-тестирование — программное обеспечение, выпускаемое для ограниченного количества пользователей. Главная цель — получить отзывы клиентов о продукте и внести соответствующие изменения.

Нефункциональное тестирование

Тестирование производительности (performance testing) — определение стабильности и потребления ресурсов в условиях различных сценариев использования и нагрузок.

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

Тестирование масштабируемости (scalability testing) — тестирование, которое измеряет производительность сети или системы, когда количество пользовательских запросов увеличивается или уменьшается.

Нефункциональное тестирование

Объёмное тестирование (volume testing) — это тип тестирования программного обеспечения, которое проводится для тестирования программного приложения с определенным объемом данных.
Стрессовое тестирование (stress testing) — тип тестирования направленный для проверки, как система обращается с нарастающей нагрузкой (количеством одновременных пользователей).
Инсталляционное тестирование (installation testing) — тестирование, направленное на проверку успешной установки и настройки, обновления или удаления приложения.
Тестирование интерфейса (GUI/UI testing) — проверка требований к пользовательскому интерфейсу.

Нефункциональное тестирование

Тестирование удобства использования (usability testing) — это метод тестирования, направленный на установление степени удобства использования, понятности и привлекательности для пользователей разрабатываемого продукта в контексте заданных условий.
Тестирование локализации (localization testing) — проверка адаптации программного обеспечения для определенной аудитории в соответствии с ее культурными особенностями.
Тестирование безопасности (security testing) — это стратегия тестирования, используемая для проверки безопасности системы, а также для анализа рисков, связанных с обеспечением целостного подхода к защите приложения, атак хакеров, вирусов, несанкционированного доступа к конфиденциальным данным.

Нефункциональное тестирование

Тестирование надёжности (reliability testing) — один из видов нефункционального тестирования ПО, целью которого является проверка работоспособности приложения при длительном тестировании с ожидаемым уровнем нагрузки.
Регрессионное тестирование (regression testing) — тестирование уже проверенной ранее функциональности после внесения изменений в код приложения, для уверенности в том, что эти изменения не внесли ошибки в областях, которые не подверглись изменениям.
Повторное/подтверждающее тестирование (re-testing/confirmation testing) — тестирование, во время которого исполняются тестовые сценарии, выявившие ошибки во время последнего запуска, для подтверждения успешности исправления этих ошибок.

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.

У **тест-дизайна** есть две основные цели:

1. Во-первых, это то, что мы должны придумать. Такие тесты, которые могли бы обнаружить наиболее серьёзные ошибки для нашего продукта.
2. Вторая цель - это минимизация количества таких тестов.

Техники тест-дизайна

Тестирование на основе классов эквивалентности (equivalence partitioning) — это техника, основанная на методе чёрного ящика, при которой мы разделяем функционал (часто диапазон возможных вводимых значений) на группы эквивалентных по своему влиянию на систему значений.

Техника анализа граничных значений (boundary value testing) — это техника проверки поведения продукта на крайних (граничных) значениях входных данных.

Попарное тестирование (pairwise testing) — это техника формирования наборов тестовых данных из полного набора входных данных в системе, которая позволяет существенно сократить количество тест-кейсов.

Техники тест-дизайна

Тестирование на основе состояний и переходов (State-Transition Testing) — применяется для фиксирования требований и описания дизайна приложения.
Таблицы принятия решений (Decision Table Testing) — техника тестирования, основанная на методе чёрного ящика, которая применяется для систем со сложной логикой.
Доменный анализ (Domain Analysis Testing) — это техника основана на разбиении диапазона возможных значений переменной на поддиапазоны, с последующим выбором одного или нескольких значений из каждого домена для тестирования.
Сценарий использования (Use Case Testing) — Use Case описывает сценарий взаимодействия двух и более участников (как правило — пользователя и системы).


САНЯ, ЧЕ ЗА ФИГНЯ? КАКАЯ ЕЩЕ ПШЕНИЦА?
ГО С НАМИ МАМОНТА ФИГАЧИТЬ
ПИВКО ВЫ ТОЖЕ ИЗ МАМОНТА ДЕЛАТЬ БУДЕТЕ?

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