ЦИКЛ ПО

1. Идея проекта - клиент приходит
2. Создание требования - бизнес аналитик и клиент
3. Анализ – вся команда начинает анализировать, для кого, что, где, на чем
4. Проектирование - архитект прокта начинает говорить как писать код, базу данных строят

И начинают разрабатывать продукт

1. Разработка – Девелп начинают писать код
2. Тестировщики – понятно ли, красиво, ищут баги, понравиться ли клиенту и осответстуют ли все это требованиям
3. Внедрение или выдача клиенту – продукт внедряем и выставляем в плаймаркет
4. Обслуживание или поддержка - 2 девелопера смотрят за сайтом, чтобы он хорошо работал

Тестировщик

На 2 стадии начинает уже тестировать требования

1. Четкие ли требовния, законченные, описаные, не противоречили друг другу, ческие сформулированые, чтобы их обязательно поняли, для маленьких и тупых
2. Начинает анализирвоать на какие системах будет тестирвоать, сколько нужно людей, что будем и что не будем проверять, какой фунционал, что будет являться критерием прохода , а что падением,т.е какой случай будет считаться что тест упал
3. Проетировка – проектирует тест план , как будет проходить тестирование , все записывает в док. После этого когда идет разработка и написание кода, тестировщик уже начинает тестировать. Как вышел кусок кода, тестировщик сразу протестировал. Это сокращает время на разработку. И быстрей их исправить. Во время внедрения проверяет работает ли эта программа на всех системах. И на фазе обслуживание, проверяет не появились ли новые баги.

ЦИКЛ тестирование.

1. Требование. См выше
2. Плагнивароние см выше
3. Анализируем все это
4. Дизайн. Создаем тест кейс, поверхностные тетсы. Описание что будет протетсировано
5. Имплементация. Детализируем тест-кейсы. Проверяем шаги
6. Выполнение теста. Какие упали, какие не упали. Если какието упали, открываем баг-репорт

Багрепорт документ, который говорит, что в программе есть ошибка., и что девелопер должен ее исправить. Открываем багрепорты, потом перетестируем эти брепорты, убеждаемся, что они были исправлены. И что больше этой ошибки нет.

1. Репортинг и выводы. Собираем все наши багрепорты в одну папку. Сохранеям все это.
2. Закрытие тестирование. Сохраняем всю эту папку до следующих релизов. , проектов.
3. Выход критерия . Формулируется на фазах вывода и закрытия тестирвоания. Когда пора прекратить тестирование. Когда закончился бюджет, когда продуктменеджера сказал стоп., когда клиент сказал стоп, когда тетсы хорошо проходят.

МЕТОЛОГИЯ,

1. Водопад, каждая ступень идет друг за другом. Работают либо старые проекты либо короткие, которые длятся год. , либо очень строгие медицинские или военные. Клиент видит только в конце результат.
2. Итерация или спринты эти требования. Будем повторять одно и тоже, менять требования, менять продукт. И т.д , тестировать, и отдаать клиенту каждый маленький кусок. (AGILE)

Подходит и для больших и для маленьких проектов. Клиент сразу видит результат. Клиент может менять требования.

Делиться на **SCRUM и KANBAN**, общее у обоих есть итерация и спринты, готовый продукт по ним работают японские компании, например, тойота, должен быть готовый продукт.(зеркало)

**SCRUM и KANBAN** есть SCRUM доска, сотоит из колонок

Baclog место где живут все карточки

В процессе, делать, ревью(должны проверить другие члены команды), готово

**SCRUM и KANBAN** есть приоритеты, размер и сложность, митинги,

**SCRUM и KANBAN отличие**

1. В скраме нужно проходить все митинги
2. В конце спринта в скраме РЕТРОСПЕКТИВА
3. В скраме есть 2 рроли: скрам-мастер и продукт-овнер

Скрам мастер – это человек, которчй отвечает чтобы вся команда следовала скраму, чтобы все участвовали в митингах , чтобы не было блокеров

Продукт-овнер это чел чаще это бизнес аналитик или продукт менеджер, знает много программ, можно с любым вопросом прийти

1. В скраме оценивание заданий обязательно
2. В скраме есть приоритеты их менять нельзя,в комбане приоритеты можно менять

Митинг daily stand up

1. Что ты делал вчера
2. Что ты будешь делать сегодня
3. Есть ли у тебя блокеры

V модель нет бизнес аналитика, сложно реализовать

Test driven development

Сначла тестировщики пишут тест на программу которого нет, потом разработчики пишут программу

Экстремальное програмирование

МИТИНГИ

1. Спринт ревью . опредееляется цель, приоритеты, и какие задания будем брать

Каждый чел определяет количество часов, который он должен сделать на спринт

1. Грув , планируется цели, приоритеты
2. daily stand up
3. Ретроспектива, подбивают итоги

-что было хорошо

-что было плохо\-что мождно улучшить

ТЕСТ-кейс

Id

Название, описание

Пред условие(список действии, например уже зарег на сайте)

Шаги

Ожидаемый рез

Пост шаги(который можем или не можем делать)

Тестовые данные

Тест стратегия – цель, план, количсетво часов, окружение. Устарело

Баги

Функциональные

Визуальные

Логические

Удобство использования

Ошибки безопасности

Ошибки перевода

Правила хорошего багрепорта

1 баг – 1 багрепорт

Все шаги максимально детально описан, не пропуская

Краткие: что, где, когда

Id

Название

Предусловие,

шаги

постусловие

Ожидаемый результат

актуальный результат

На кого назначен(имя разработчика)

Вложение(скриншоты, видео, логи)

Приоритет

степень влияния(на сколько сильно дефект влияет на систему) Высокий, средний, низкий, космтический

Категория, тип дефекта

статус

Система, версия ос,

Дата открытия,

дата закрытия

jing для скриншотов багов

jira

Типы тестирования

Статистическое и динамичсекое

Статистическое проверяет всю документацию(чек листы, тест кейсы, багрепорты, требование, тест план, тест стратегия, код програмистов)

Динамическое – тестирование программы

Подходы тестирование проактивное заранеее тестировать и реактивное в момент

Ручое и автоматизированнок

Позитивное и негативноке

Белый , чернн, сер ящик

Статис и динамич тест

Вериф и валид

Статическое равно верификации, динамич равно валид

Уровни тестирования

1. Юнит, модуль компонет – маленький кусок программы
2. Интеграционные – взаимодействие 2 модулей(добавить вкорзину, и нажать на 1+) (модульное интеграционное тестирование и системное интеграционное тестирвоание)
3. Системные тесты – интернет магазин
4. Эсептинг тес(приемо-сдаточное)т, тестируют программу чтобы ее приняяли по минимальным требованиям
5. Альфа тесты, ее тестирует команд аразработчиков
6. Бета тестирование, тестирование людьми не относящиеся к команде, людьми с улицы

Интеграционное тестирование

1. big bang testing все модули свзяна одновременно, получается полная система
2. Top down integration – тестируются все высоуровненвый мдуди
3. Bottom up – снизу вверх

ТЕСТ ДИЗАЙН

1. Эквивалентное разделение(валидное и невалидное)
2. Техника граничных значений (валидное и не валидное, пароль от 5-10, 4, 5, 10, 11)
3. Таблицы и принятий решений
4. Парное тестирование
5. Диаграммы ( state-transition diagram, use-case diagram(диаграмма ролей)) перехода состояний
6. Техники связанные с опытом , угадывание ошибки error guessing

-Сильно увеличить и сильно уменьшить экран,

-деление на 0

-ввести пустое знаечние

-негативное тестирование дат, 30 февр

- пустые файлы техстовые файлы на аватапку, на изображение,

-в поле телефона вводим текст

7. Иследовательское тесирование

-тестирование без знания продукта

-тестирование без документации

- тестирование с нехваткой времени

Ad-hoc тестирование без документации, просто клики

**Принципы тестирвоания**

1. Тестирвоание не гарантирует отсутствие дефектов
2. Исчерпывающее тестирование невозможно
3. Ранее тестирование
4. Групировка дефектов
5. Парадокс пестицидов, нужно постоянно обновлять набор тест-кейсов
6. Тестирование зависит от контекста
7. Отсутствие дефектов обманчиво

ГИТ

1. В Гит баш, вводим NUL .gitconfig
2. Открываем файл через блокнот
3. Вводим инфу о пользователе

Sourcetree скачать для гит гуи

Тестовое покрытие – это ко-во тестов / ко-во требований Если т.покрытие > 1 это неплхо, если >2 то хорошо, если <1 то значит плохо, и нужно больше создавать тест кейсов

Плотность дефектов в модуле = ко-во дефектов в модуле / общее ко-во дефектов \* 100%

Часть повторно открытых багов = ко-во повторно открытых багов / ко-во всех багов

Средний час жизни дефекта = время потрач на все баги / общее ко-во багов

Xpath – локатор который помогает искать элементы в html css

1. Абсолютный Xpath
2. Относительный

//h2// или //\*//h1

//section[@class = ‘gallery’]/h2 по атрибуту класс

[contains(@class = ‘’)] если поиск по длинному класу

//h2[text() = ‘’Кадры’’] поиск по тексту или

H2[Contains(text(), ‘дры’)]

//h2[normalize-space() = ‘….’] чистит лишние пробелы

Идентификация, это когда система спрашивает кто ты

Аутентификация - вопрос докажи что ыт это ты

Авторизация- это когда система понянла что ты это ты, и дает тебе фойти и все права

Тонкий клиент – мы не делаем исчисления, и переносим всю информацию на сервер

Толстый клиент – который может делать исчисления, обработку, хранения информации самостоятельно, 1С

Вопросы собесед

1. Типы тестирования
2. Разница между верифи и валидац
3. Принципы тестирования 7шт

-тестирвоание покаывает наличие багов

-исчерпывающее тестирвоание невозможно

-ранее тестирвоание

-накопление дефектов(оснвоное ко-во дефектов находится в одном модуле)

-парадокс петицидов (надо составалять новые и новые тест-кейсы)

-тестирование зависит от контекста

-заблуждение об отсутсвии ошибок

4. разница между load, performance , stress тестировании. Load проверка пведение программы при нагрузке (20тыс пользоваттелей заходят в лк) . Performance измерение отклика программы при нагрузке, за сколько миллисекунд перейдет на другую страницу. Stress нагрузкой убиваем программу

5. Разница между степенью важностью(насколько ломает систему, возможность работать) и приоритетеом(насколько срочно нужно исправить, влияет на репутацию)

6. статусы коды http

7. техники тест дизайна

8. scrum . У него есть роли(мастер-мастер и продукт owner), спринты(маленькая итерация, длиной от 1нед до 3нед, меся, к конце долдны сделать кусок готовой работы), доска, приоритеты меняются либо до спринта либо после, но не во время, в скраме нужновзять себе задание, в скраме важно ко-во людей от 3-до 8

9. Из чего состоит тест кейс: id, название, пред условий, шаги, пост условия, ожидаемый результат, скриншоты, тестовые данные

10. из чего состоит бег репорт: id, название, описания, предсостояния, шаги, постсостояния, степень важности, приоритет, тип дефекта, кто его открыл, когда его открыли, на кого этот дефект назначили, тестовые данные, скриншоты, видео, логи

11. Цикл тестирование: требования, планирование(типы тестирования) и анализ, дизайн(тест-кейсы) и контроль, имплементация, (детализируем), выполнние тестов, выводы (табличка(id, название, результат), предоставляем продуктменеджеру, клиенту), из плохих тестов открываем багрепорт, их фиксят, мы перететсируем, закрытие тсета: собираем всю инфу, сохраняем, чтобы не допускать новых ошибок

12. в чем разница между черн, белым и сер ящи

13. типы запросов api : get – получаем, post – создаем, put – изменяем полносью все , delete path –изменяем, только те паранметры кторый нужно поменять

14. Типы join: left, Right, inner, full

15 связи с бд: один к одному, один ко многим, много ко многим

16. разница между qa и qc