3. Разработка

3.1 Разработка::Подготовка спринта

- Техническое задание (если есть) разбито на фичи, **фичи — помещены в backlog** (список фич по приоритетам).
- Список фич/историй сформирован **полностью на этап**
- Backlog — храним в **таблице** Google Docs. Доступен команде. Редактирует менеджер. **Отсортирован по приоритету**

	ПРИОРИТЕТ	**ФИЧА/User Story**	**КРИТЕРИИ ПРИЕМКИ / ПРИМЕЧАНИЯ**	**...**
№	Уникальное число	Фичи, а не задачи	Тесткейсы.	

- Фичи отсортированы по приоритету.
- Тесткейсы понятны, **полезны** и охватывают максимальное количество классических и экстремальных случаев! Можно привлекать QA-специалиста. В крайнем случае можно написать кейсы после планинга.

Менеджер <u>четко понимает все фичи</u> бэклога. Не только “что надо сделать”, но и “<u>как это будет</u> выглядеть и <u>работать</u>”.

- Спринты — 1-2 недели *в зависимости от опыта команды.*
- Новых задач после планинга в спринт не брать.

3.2 Разработка::Планирование

- Создать чат команды по проекту.
- Назначить **время и место** для проведения планинга. **Оценки делать ВСЕЙ командой.**
- **Длительность планинга порядка 90 минут.**
- Уведомить команду о времени и месте. Обеспечить явку.
- Заблаговременно **до планинга** дать команде изучить бэклог и ТЗ (только те фичи, которые войдут в спринт). **Выяснить все неясные вопросы.**
  - Убедиться, что команда **прочитала ТЗ.**
  - **Убедится, что команда ПОНИМАЕТ ТЗ (задает вопросы).**
  - Дать время на планирование архитектуры. (*В очень сложных проектах вынести архитектуру на отдельный этап*

  - ***Архитектуру делает один человек***

- Собрать команду на планинг, с выключенными телефонами.
- Создать спринт внутри проекта.
- Последовательно читать фичи. Разбивать на технические задачи.

- *Задачи ставить в SCRUMBAN в **однозначной формулировке. Четко. Ясно. Выполнимо.***

  - Техническая реализация. Должны быть понятна всем и

принята однозначно!
    - Критерии приемки.
        - Запланировать QA на написание тест-кейсов.
- Если есть ТЗ — **КАЖДОЕ слово** понимать, **подкрашивать маркером**, **копировать текст** в SCRUMBAN.
- Оценить с помощью Planning Poker поставленную задачу:
    - Сроки не зажимать.
    - Укладываться в бюджет с резервом.
    - в SCRUMBAN писать оценку, которую дала команда.
- Назначить End Date (закончен код) и Release Date (сдача этапа):
    - Сообщить команде.
    - Проставить в SCRUMBAN.
    - Укладываться в бюджет.
    - Закладывать буферы времени, относительно оценок.
- Спланировать работу на сегодня.
- **Назначить время и место стэндапов.**
- В **теме чата** прописать **даты окончания, релиза и время стендапа.**
- В календари разработчиков ставим продолжительность этапа **с учетом резерва** на тестирование и рисков на сложность задачи и скорость работы команды.
- Вместе с одним из членов команды и QA подготовить приемочные тесты/критерии приемки/тест-кейсы.

3.3 Разработка::Standup

- **РИТМ:** Проводить в **одно и тоже время** в **одном и том же месте.**
- *Команда должна самостоятельно подготовиться к стендапу: актуализировать статусы задач на доске, в обозначенное время стендапа быть рядом с рабочим местом менеджера. Если задачи не актуализированы или на стендап кто-то опаздывает — тому по башке!*
- Собрать команду. На один монитор вывести проект, на второй - канбан спринта. Стендап проводится у компьютера МЕНЕДЖЕРА.
- Последовательно каждому члену команды задать **три вопроса**. Соблюдать очередность.
  - **— Что было сделано вчера?**
  - **— Что будет сделано сегодня?**
  - **— Какие есть проблемы?**
- **В дискуссии не вступать.** Параллельные потоки пресекать! В проблемы ВРУБАТЬСЯ и затем — РЕШАТЬ! Не врубился? Эскалируй!
- Технические темы выносить за рамки Standup.
- Не более **3-х минут** на человека.
- Требовать рассказывать **своими словами** об изменениях в системе, *а не о номерах задач.*

- Настаивать на демо готовых фич на экране; Пусть **покажут пальцем, что готово**. И чтоб было понятно. Не вестись на “абстракции” в коде.

3.4 Разработка::Ежедневная работа

- Разработчикам — сразу после завершения задачи проверять и отмечать тесткейсы. PM — обеспечить это!
- **Еженедельно** отправлять клиенту **отчет** о ходе работ.
- По договоренности — дать доступ к баглистам/backlog/scrumban (на чтение!)
- Управлять ожиданиями.
- На КРУПНЫХ проектах архитектуру баз данных и инфоблоков делать заранее и проверять руководителем
- Напоминать команде, что **главное — качественный (ахуенный) проект.**
- Периодически **просматривать систему.**
- Как можно раньше внести **тестовый контент**, максимально приближенный к реальному
- **Устранять "затыки" у команды,** решать открытые вопросы.
- Обеспечивать дизайном, версткой, помогать команде.
- Следить за аномалиями. Не ссать ЭСКАЛИРОВАТЬ! Не врать и не бояться ;)

3.5 Разработка::Тестирование

3.5.1. Организация процесса тестирования внутри

- В начале спринта составить с **разработчиком и QA тест-кейсы на каждую задачу.**
- **QA готовит чеклисты из типовых чеклистов, для проверки каждого соответствующего элемента и размещает их на отдельной вкладке в google doc Дневника Проектов.**
- При необходимости ДО тестирования провести **code review** и правку по code review (*На не сработавшихся командах проводить 1 раз в спринт или чаще. На опытных — раз в 2-3 спринта).*
- Первая проверка — **QA и разработчик совместно:**
  - следить, чтобы **разработчики реально тестировали** (а не сразу фиксили баги, как находили). Тест разработчиками означает, что они проверили и формально покрасили чеклисты, подготовленные QA.
  - отсматривать **формальное выполнение тест-кейсов.**
  - разработчики проходят по чеклистам, реально проверяют каждый тесткейс и красят его. **Именно это означает, что “разработчик за собой проверил”.**
- Результат тестирования — в таблицу **Google Docs**, в тот же файл, где лежит backlog, **или карточками в SCRUMBAN.**
  - Прислать ссылку на багрепорт (на чтение) — **заказчику** с уведомлением о старте тестирования. Это снимет с тебя вопросы “вы плохо тестировали”.
- Отсортировать результаты теста по группам:

  0 — Критические баги.

  1 — Критичное Usability, забытые фичи.

  2 — Некритичные баги.

3 — Некритичное Usability.
4 — Тексты.
8 — Хотелки, не будем делать.

- Если багов много — организовать работу (канбан).
  - Пусть команда проставит **оценки багов.**
  - Показывать программистам за раз — только **часть багов** (не более 2-х экранов, самые приоритетные)
  - *Ежедневно удалять на отдельный, недоступный для программистов лист — пофиксенные и проверенные баги*
  - Новые найденные баги заносить на отдельную вкладку
  - После завершения работы над одной вкладкой — просить программистов проверить систему полностью. Время не зажимать
  - К найденным ими багам добавлять свои.
  - Обеспечить итеративность.
- Внести реальный контент.
- Протестировать проект лично. В сложных случаях — включая админку.
- Провести тестирование внешним менеджером.
  - Показать проект дизайнеру, рисовавшему проект.
  - Показать проект артдиректору.
  - Показать проект Вове.

В случае реальной угрозы **протрахаться с багфиксом** менеджер обязан 1. получить оценку багфикса, 2. выстроить приоритеты и контрольные точки, 3. декомпозировать большие задачи, 4. оперативно отвечать на возникающие вопросы, 5. своевременно информировать о затыках, 6. привлекать (по согласованию) других разработчиков на помощь в решении проблем.

3.5.2. Дать знать заказчику — тестирование снаружи

После того, как стартовало тестирование, необходимо отправить заказчику письмо с информацией об этом.

Цель: держать в курсе происходящего. Еще раз показать прозрачность процессов в студии. Предотвратить возможный негатив в стиле «А как же вы так тестировали?! БАГ НА САЙТЕ». Важно: доступ к документу давать **только на чтение**.

Письмо в следующем стиле:

Николай, сегодня мы стартуем тестирование вашего проекта. Напомню — в текущий спринт вошло вот это, вот это и еще вот то.

Вкратце о процессе тестирования в нашей студии

- Верстка тестируется на соответствие макетам, плюс мы проверяем, как сверстанные макеты отображаются в разных браузерах и на разных устройствах.
- При планировании разработки мы составляем специальные тест-кейсы к каждой задаче. Именно по этим тест-кейсам разработчики проверяют корректность выполнения.
- После окончания спринта тестировщик также проходится по всем тест-кейсам и отмечает выполненное, попутно фиксируя найденные ошибки.
- Сайт в обязательном порядке просматривается менеджером проекта. Менеджер фиксирует все найденные ошибки и неточности.
- Программисты исправляют баги. Проводится повторный тест.
- Заключительный этап — тестирование внешним менеджером, никак не касавшимся сайта ранее. Он смотрит, насколько удобен сайт для конечных пользователей.
- Предварительно мы протестировали верстку сайта на предмет соответствия макетам и корректного отображения в браузерах.

О процессе тестирования в картинках

Все комментарии фиксируются в гугл-документе: https://docs.google.com/spreadsheets/d/1fdYNKRd5c9za1qde0tlTghlE15kdjGpZKo37nTgY-78/edit#gid=765416764. Процесс тестирования и отладки в среднем занимает 2–4 рабочих дня.

Николай, в этом документе вы можете наблюдать за процессом тестирования: у вас есть доступ на чтение. Документ внутренний, поэтому в нем возможны неформальные комментарии :)

Удачного дня!

3.6 Разработка::Демонстрация

- **Согласовать время** разговора за 1 день до сдачи.
- За 1 час до сдачи — **удостовериться, что все в силе.**
- **Демо проводить совместно с командой.** *Каждый должен чувствовать свою ответственность и причастность.* Если это неприемлемо — согласовать с Вовой.
  - Использовать видео-канал.
- Обязательно **показывать и рассказывать голосом.**
- **Если канал позволяет — показывать сайт со своего экрана по skype.**
- Подготовить и настроить клиента на **позитив и принятие проекта.**
- Рассказать что команда проделала много работы, вкратце перечислить, что именно.
- Дать ссылку на проект.
- Пройти по всем основным меню, рассказать, как было сделано и почему.
  - Обосновать принятые решения.
- Получать обратную связь. Фиксировать предложения.

Закончить на позитиве.

- Законспектировать разговор.
- **Незамедлительно отправить callreport (cc Вове)**
- Если обнаружат мелкие баги на демо — **устранить незамедлительно и тут же сдать**, не передавая команду на другой проект и до проведения ретроспективы.
- **ПРЕВОСХОДИТЬ ОЖИДАНИЯ!** Стараться **сделать клиенту небольшой бонус** (больше чем обещали). Например,
  - — Уложиться раньше срока
  - — Сделать пару дополнительных фич
  - — Добавить пару фишек
  - — Внести контент

  Акцентировать внимание клиента на том, что это было сделано бесплатно, как подарок.
  Отметить это же в коллрепорте.
- Отжать акт и постоплату (если есть).
- Назначить время для проведения планирования с клиентом следующей итерации.

3.7 Разработка::Ретроспектива

Видео по теме. К просмотру обязательно!

- Проводить **сразу после завершения спринта** (до погружения команды в другой проект. Может быть даже ДО ДЕМО).
- **Назначить время.** Согласовать с командой.
- Попросить команду заранее повспоминать плюсы и минусы этапа.
- Собраться без лишних ушей.
- **Настроить команду на конструктив.** Привет. Как дела?

Цель: улучшить наши процессы. Напомни ее в самом начале!

- Последовательно каждого члена команды спрашивать о плюсах и минусах этапа.
- В споры не вступать. Модерировать. Параллельные потоки закрывать.
- Плюсы, минусы и идеи — записывать на стикерах, вывешивать по 4-м квадрантам (+-*/)
- Проанализировать минусы. **Сформировать идеи** по их **решению** (любые).
- В случае сложных проблем — использовать rootcause-анализ.
- **СТАРАТЬСЯ ВЫЯВИТЬ ПРОБЛЕМЫ. НЕ ССАТЬ!**

- **Идеи не оспаривать.**
- Сформировать план из **реальных конкретных действий**. Никаких “в следующий раз работать лучше” или “Всегда быстро отвечать на Skype”. **Не процессы, а конкретные, проверяемые, конечные, выполнимые действия!**
- Назначить ответственных по каждому пункту.
- После ретроспективы задачи поставить в SCRUMBAN в проект RETRO или следующий спринт на исполнителей. Стикеры вернуть на доску:
  - Основная задача содержит дату ретроспективы.
  - Принятые решения содержать — помещены в подзадачи.
  - **Запланировать** с Вовой **в Автоматизатор время на выполнения решений по ретроспективам.**
  - По возможности сделать задачи с ретроспективы до начала следующего спринта или включить их в спринт.
- Следующую ретроспективу начать с анализа сделанного. Развесить старые стикеры по 4-м квадрантам (+-/*).