Работа с требованиями

NX Bootcamp

User story

User story

User Story — это короткая формулировка намерения, описывающая что-то, что система должна делать для пользователя.

User story - ограничения

- отсутствие деталей, ведет к различным интерпретациям
- могут плохо масштабироваться на больших проектах
- требуется большая компетенция от разработчика
- не учитывают полного бизнес-процесса и, как результат, в итоге могут оказаться ошибочными из-за отсутствия всестороннего исследования предметной области.

User story - преимущества

Привязка к малым, реализуемым и проверяемым частям функциональности, облегчает быстрый выпуск релизов и частые отзывы клиентов.

Легкое понимание заинтересованными сторонами.

Простой формат, чтобы люди могли научиться писать их за несколько минут, внимательно отслеживая, что представляет ценность для бизнеса.

Инициируют переговоры для последующей декомпозиции и исследования.

User story - преимущества

Привязка к малым, реализуемым и проверяемым частям функциональности, облегчает быстрый выпуск релизов и частые отзывы клиентов.

Легкое понимание заинтересованными сторонами.

Простой формат, чтобы люди могли научиться писать их за несколько минут, внимательно отслеживая, что представляет ценность для бизнеса.

Инициируют переговоры для последующей декомпозиции и исследования.

User story - формат

Я как [роль]
Хочу [функционал]
Для того чтобы [выгода]

Как <роль>, мне необходимо <поведение>,
чтобы получить <ценность бизнеса>

Для того чтобы получить <ценность бизнеса>,
мне как <роли>, необходимо <поведение>

User story - ошибки

«Как пользователь я хочу управлять рекламными объявлениями, чтобы удалять устаревшие или ошибочные объявления»

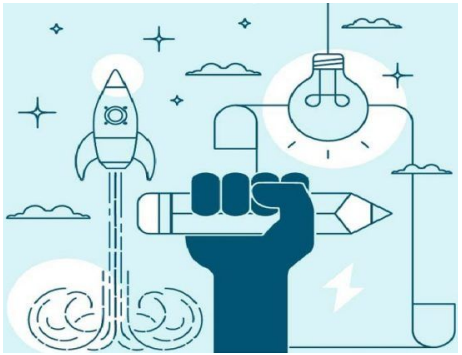
«Как продакт оунер, я хочу, чтобы в системе была возможность удалять объявления, чтобы юзеры могли удалять объявления»

«Как девелопер, я хочу заменить виджет папок, чтобы у меня был лучший виджет папок»

User story - рекомендации

- Лучше написать много историй поменьше, чем несколько громоздких.
- История в идеале должна быть написана избегая технического жаргона.
- Истории должны быть написаны таким образом, чтобы их можно было протестировать.
- Тесты должны быть написаны до кода.
- История должна выполняться без привязки к конкретным элементам UI.
- Каждая история должна содержать оценку.
- История должна иметь концовку — т.е. приводить к конкретному результату.
- История должна помещаться в итерацию.

Практическое задание 017



Каждому слушателю написать минимум 5 User story для одного из действующих лиц системы (пользователей) (практическое задание 002).

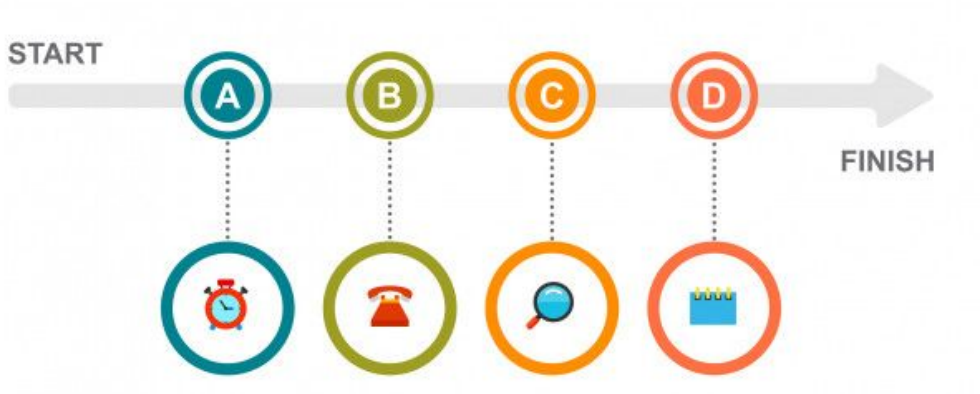
№№	User story	Источник (ФИО, контакты)
1	Я как кладовщик, хочу иметь возможность видеть количество груза которое приедет на склад в следующей партии, для того чтобы вызвать нужное количество грузчиков, чтобы закончить разгрузку вовремя.	
2	Я как логист, хочу иметь возможность видеть время отправления машины, для того чтобы иметь возможность ускорить процесс загрузки.	

Use case, вариант использования, пользовательский сценарий

Use case

Вариант использования — это описание того, как пользователь может взаимодействовать с системой, чтобы достичь определённой цели.

Каждый вариант использования представляет собой последовательность простых шагов, которые пользователь должен пройти, чтобы достичь цели.



Use case - зачем нужны

Техника выявления требований

Техника проектирования взаимодействия

Метод раскрытия объёма функции

Формат представления и группировки требований

Единица планирования и сдачи системы

Основа для тест-кейсов и документации

Use case - виды потоков действий

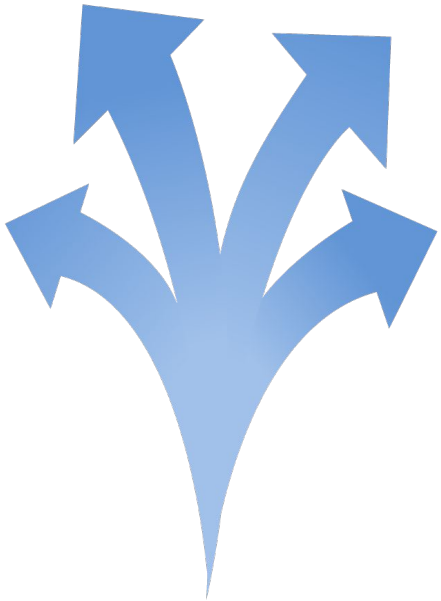
Система предоставляет данные и возможности

Пользователь сообщает данные и даёт команду

Система убеждается в выполнении условий

Система выполняет операции над данными

Система сообщает об успешном результате пользователю



Use case

Правила проектирования взаимодействия

- Система должна подсказывать следующий шаг
- Пользователь не может выполнять операции,
  он только даёт данные и команды
- Система должна сообщать о результатах операции

Use case - предусловия

Отражают логические правила, которым должен удовлетворять сеанс пользователя

Пример:

Редактор просматривает список правок

Редактор решает отменить одну из правок

Use case - триггер

Событие, которое запускает выполнение сценария

Примеры:

Курьер даёт команду на открытие списка заказов

Система получает уведомление о превышении квоты пользователя от стороннего сервиса

Use case - постусловия

Условия, которые должны быть выполнены после успешного завершения юскейса.

Пример:

Комментарии Пользователя сохранены в системе.

Use case - альтернативные потоки

Необходимы для управления ходом выполнения сценария:

- Ветвления, вариации
- Циклы
- Обработка ошибок

Альтернативный поток

- Условие входа
- Сценарий обработки

Передача управления назад / в другую точку:

…

5а. Пользователь не указал имейл:

5а1. Система сообщает пользователю, что он не указал имейл.

5а2. Выполнение сценария переходит к шагу 3.

Альтернативный поток - ветвление

…
4. Система предлагает отправить сообщение SMS или встроенным сообщением
5. Пользователь запрашивает отправку сообщение SMS.
…

Альтернативный поток (для вариантов исполнения):
5а. Пользователь запрашивает отправку встроенного сообщения:
    5а1. <Организация отправки встроенного сообщения>
    …
    5аN. Выполнение сценария переходит к шагу 9.

Альтернативный поток - циклы

…
(Шаги 1-3).
4. Система убеждается, что в списке есть необработанные сообщения.
5. Система предлагает обработать следующее сообщение или закончить обработку.
6. Пользователь даёт команду на обработку следующего сообщения.
… (обработка сообщения)
10. Сценарий продолжается с шага 1.
…

Альтернативный поток (для цикла):
4a. Необработанных сообщений нет:
    4a1. Система сообщает пользователю, что необработанных сообщений нет.
    5a2. Сценарий завершается.

Use case - ошибки

…
3. Система просит указать … и имейл
4. Пользователь вводит …
5. Система убеждается, что пользователь указал имейл в правильном формате
…
Альтернативный поток (обработка ошибок):

5а. Пользователь не указал имейл:
    5а1. Система сообщает пользователю, что он не указал имейл.
    5а2. Выполнение сценария переходит к шагу 3.

5б. Формат имейла ошибочный:
    5б1. Система сообщает пользователю, что в имейле есть ошибки.
    5б2. Выполнение сценария переходит к шагу 3.

EARS сценарии

The Easy Approach to Requirements Syntax

Состав:

- Описание
- Входные параметры
- Логика обработки
- Выходные параметры:
- Перечень ошибок/исключений:
- Блок-схемы, примеры

Формат:

[необязательное предварительное условие] [необязательный триггер события] система должна [ожидаемая реакция системы]

EARS сценарии - виды

Безусловное поведение – определяет поведение системы, которое должно работать все время (фундаментальное требование, по факту очень редкое)

система должна [ожидаемая реакция системы]

По событию – требование работает как реакция на событие, обнаруженное на границе системы/модуля

когда [триггер события] система должна [ожидаемая реакция системы]

EARS сценарии - виды

По состоянию – требование к системе, которая находится в определенном состоянии

пока [определенное состояние] система должна [ожидаемая реакция системы]

По условию – требования к системе, которая обладает конкретными свойствами

тогда как [свойство предусмотрено] система должна [ожидаемая реакция системы]

EARS сценарии - виды

Нежелательное поведение – требования к системе, поведение которой отклоняется от нормального (похоже на требование по событию)

если [не обязательное предварительное условие] [необязательный триггер события] система должна [ожидаемая реакция системы]

Коберн - вариант использования полный

Название: <название должно быть в виде краткой фразы с глаголом в неопределенной форме совершенного вида и отражать цель>

Контекст использования: <более длинное предложение цели, при необходимости условия ее нормального достижения>

Область действия: <область действия проектирования, в которой разрабатываемая система рассматривается как “черный ящик”>

Уровень: <обобщенный, цели пользователя, подфункции>

Коберн - вариант использования полный

Основное действующее лицо: <ролевое имя для основного действующего лица или описание>

Участники и интересы: <список участников и ключевых интересов в данном варианте использования>

Предусловие: <то, что, как ожидается, уже имеет место>

Минимальные гарантии: <как защищаются интересы участников при всех исходах> Гарантии успеха: <что имеет место, если цель достигнута>

Коберн - вариант использования полный

Основной сценарий:
<шаги сценария от триггера до достижения цели и завершение>
<номер шага> <описание действия>

Расширения:
<расширения, каждое из которых обращается к шагу основного сценария>

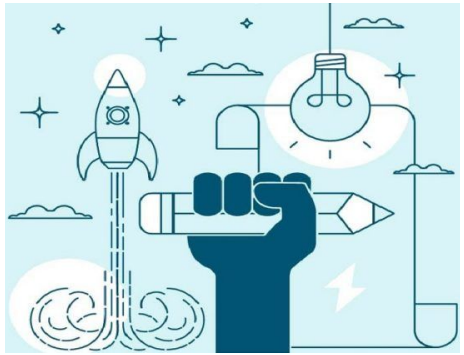
<изменяемый шаг> <условие>: <действия или подчиненный вариант использования>

Коберн - вариант использования табличный

Клиент	Система
Вводит номер заказа	
	Определяет, что номер заказа соответствует выигравшему номеру месяца.
	Регистрирует пользователя и номер заказа как победителя данного месяца.
	Посылает электронной почтой сообщение менеджеру по продажам.
	Поздравляет клиента и инструктирует его, как получить приз.
Выходит из системы	

Практическое задание 018

Каждому слушателю написать 3 варианта использования системы на основе реестра UseCase (практическое задание 016).

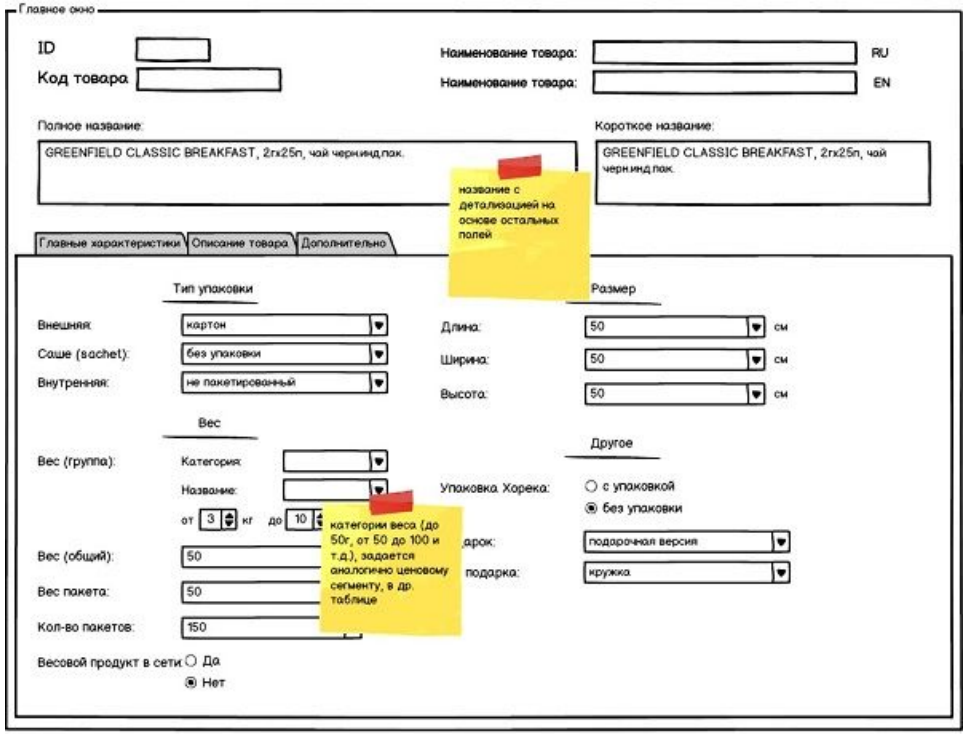


Прототипирование интерфейсов

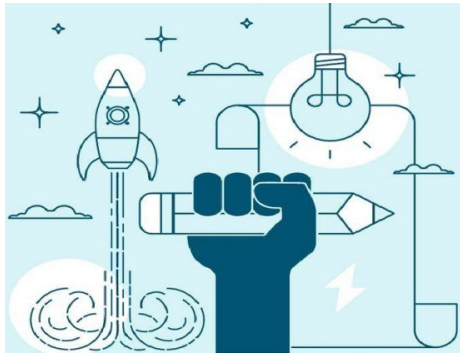
Прототип интерфейса

Базовые правила создания интерфейсов:

1. Доступ к любой функции должно выполняться максимум за два клика
2. Структурируйте информацию по важности слева направо и сверху вниз



Практическое задание 019



019 Разработать прототип интерфейса соответствующего одному из описанных в задании 018 вариантов использования.

Заполнить таблицу описания прототипа интерфейса

Нефункциональные требования (характеристики качества)

Нефункциональные требования - виды

Нефункциональные требования, фактически являются характеристиками качества ПО, и влияют на архитектуру системы.

Виды нефункциональных требований:

- первая группа (run-time) – требования, относящиеся ко времени работы приложения или системы;
- вторая группа (design-time) – требования определяющие ключевые аспекты проектирования приложения или системы.

Нефункциональные требования - run-time

- **Доступность** — определяет время непрерывной работы. Измеряется максимально допустимым временем простоя.
- **Надежность** — требование, описывающее поведение приложения или системы в нештатных ситуациях (автоматический перезапуск, восстановление работы, дублирование данных, резервирование логики)
- **Время хранения данных**
- **Масштабируемость** — требования к горизонтальному и/или вертикальному масштабированию. Вертикальное масштабирование направлено на повышение производительности системы. Горизонтальное масштабирование, помимо производительности, позволяет повысить отказоустойчивость системы.

Нефункциональные требования - run-time

- **Удобство использования** (с точки зрения пользователя)
- **Простота поддержки**
- **Требования к безопасности** - разграничением доступа, работа с приватными данными, снижение рисков от внешних атак.
- **Требования к производительности** - количество одновременно работающих пользователей, обслуживаемых транзакций, время реакции, продолжительность вычислений и т.д.
- **Технические ограничения** - при которых приложение должно эффективно выполнять возложенные на него задачи (объем доступной памяти, процессорное время, дисковое пространство, пропускная способность сети)

Нефункциональные требования - run-time

Требования к конфигурируемости взаимодействия и расположения компонентов:

- конфигурируемость на основе параметров, модификации путем изменения значений параметров;
- конфигурируемость на основе базовых объектов, модификации путем перекомпоновки процессов, сущностей и служебных процедур;
- конфигурируемость путем реализации новых базовых объектов, обеспечивается расширение набора процессов и сущностей;
- конфигурируемость путем новой реализации системы, когда система должна устанавливаться и настраиваться с нуля.

Нефункциональные требования - design-time

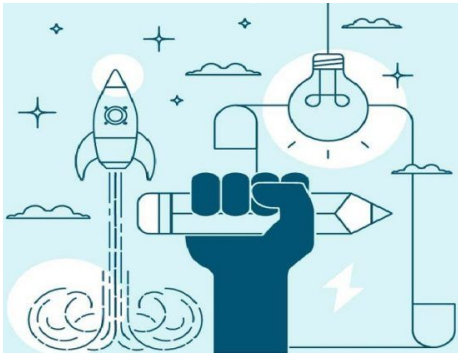
- **Требования к повторному использованию реализации или компонентов**
- **Требования к расширяемости в связи с появлением новых функциональных требований**
- **Требования к переносимости на другие платформы**
- **Требования к взаимодействию** между компонентами решения, и внешними компонентами, использование стандартных протоколов и технологий взаимодействия

Нефункциональные требования - design-time

- **Требования к поддержке** - дешевизна и скорость разработки, прозрачность поведения приложения, простота анализа ошибок и т.д.
- **Требования к модульности** приложения или системы
- **Требования к возможности тестирования** - требований к возможности и качеству автоматического и ручного тестирования.
- **Требования к возможности и простоте локализации**

Практическое задание 020

Сформулировать нефункциональные требования к программному продукту.



Thank you

NX Bootcamp

Join us!

VK

WhatsApp

Telegram

nxbootcamp@nexign.com

NX Bootcamp