Дисциплина «Проектирование веб-систем»

Лабораторная работа №4. User Story и Use-Case

Цель лабораторной работы

Определить функциональные требования к системе. Описать типичные взаимодействия между пользователями системы и самой системой и предоставить описание процесса её функционирования.

ЗАДНИЕ 1:

1. Создайте UseStory для своего продукта
2. UseStory – это по сути описание Прецедента со стороны какого-либо Актера.
3. Для каждой история задаться номер. US01, US02 и т. д.
4. Описание:
5. 1. Кто (от какого лица история)
6. 2. Что (Какие требования у актера к системе.)
7. 3. Для чего
8. Детализируйте UseStory
9. Упорядочите и Сгруппируйте UseStory

ЗАДАНИЕ 2.

1. Определяем Актеров (Actor).
2. Составляем единый список возможностей (функций) системы.
3. Разделяем их по смыслу на группы.
4. Выделяем сценарии (прецеденты) UseCase.
5. Описываем прецеденты (UseCase).
6. Строим диаграммы UseCase.

Вне зависимости от методологии разработки, которую вы применяете, первым этапом разработки будет являться формулировка требований к продукту. Набор требований к продукту представляет собой техническое задание, при этом требования делятся на функциональные (то, что система позволяет сделать, желаемая функциональность) и нефункциональные (требования к оборудованию, операционной системе и т. п.). В языке UML для формализации функциональных требований применяются диаграммы использования (use-case).

Однако первым этапом стоит отметить написания Пользовательских историй (User Story) – как части сбора требований к продукту.

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

User Story – это не требования!!!

Особенности User Story:

• являются детальным описанием требований (то-есть того, что система должна бы делать), а представляют собой скорее обсуждаемое представление намерения (нужно сделать что-то вроде этого)

• Они являются короткими и легко читаемыми, понятными разработчикам, стейк-холдерам (заинтересованные лица) и пользователям

• Они представляют собой небольшие инкременты ценной функциональности, которая может быть реализована в рамках нескольких дней или недель

• Они относительно легко поддаются эстимированию, таким образом, усилия, необходимые для реализации, могут быть быстро определены

• Они не занимают огромных, громоздких документов, а скорее организованы в списки, которые легче упорядочить и переупорядочить по ходу поступления новой информации

• Они не детализированы в самом начале проекта, а уже более детально разрабатываются «точно в срок», избегая таким образом слишком ранней определенности, задержек в разработке, нагромождения требований и чрезмерно ограниченной формулировки решения

• Они требуют минимум или вовсе не требуют сопровождения и могут быть безопасно отменены после имплементации

Структура User Story

Текст самой юстории должен объяснять роль/действия юзера в системе, его потребность и профит, который юзер получит после того как история случится USnn. Как, <роль/персона юзера>, я <что-то хочу получить>, <с такой-то целью> .

Пример:

«Представьте, что вы составляете „пожелание пользователя Amazon.com«. Пробный вариант выглядит так: „Мне как потребителю нужен крупнейший в мире магазин книг, где я могу купить любую книгу в любое время«. Это описание вполне отвечает характеру Amazon, но история получилась слишком расплывчатой, чтобы с ней можно было что-то сделать. Нужно фрагментировать нашу историю. Сделать ее действительно очень конкретной и функциональной. Приведу несколько образцов пользовательских историй, которые вы можете написать, имея в виду книжный интернет-магазин:

• *Как потребителю мне удобно искать книги по жанрам, чтобы быстро найти те, которые я люблю читать.*

• *Как потребитель я, отбирая книги для покупки, хочу сразу класть каждую в корзину.*

• *Как управляющий по выпуску новой продукции я хочу иметь возможность отслеживать покупки наших клиентов, чтобы быть в курсе, какие книги им можно предлагать.*

T. e. User Story желательно должен соответствовать следующим критериям:

1. Есть один **actor**
2. Есть одно **действие**
3. Есть одна **ценность** / value / impact.

Actor:

Определите роли в Системе например — Пользователь, Оператор и Админ. И креативите по 100 историй, которые начинаются как «Как Пользователь Я ...».

Джеф Паттон предлагает следующее:

1. Разделите всех актеров на группы. Целевая группа, важная группа, менее важная группа и т. п.

2. Дайте уникальные названия актерам в этих группах. Даже если в системе у них будет одинаковые роли «Пользователя системы».

3. Пишите истории с точки зрения этих актеров указывая их уникальные названия.

4. В результате вы сможете визуально увидеть какие истории необходимы для актеров целевой группы, какие – для каждой группы и тп. Вы не просто можете использовать это при разборе истории и выстраивания анализа вокруг указанного актера. Вы сможете более правильно выстроить приоритет, так как истории актеров целевой группы для нас более важны.

Действие:

Действие – это суть истории, «что нужно сделать». Что можно улучшить. Действие должно быть одно – основное. Нет смысла описывать «авторизуется и выполняется поиск» или «указывает параметры поиска и выполняет поиск». Укажите то действие, что вам действительно нужно.

Важно описывать историю на уровне «ЧТО?» делает, а не «КАК?» Это главное в истории. Опишите проблему, а не ее решение.

Ценность:

Главная проблема с User Story. Все должно быть согласно шаблону, и потому вы пишите «чтобы ...» и какую-то чушь, в которую сами не верите. Уберите эту часть из истории. Если ничего не потеряли — значит формализация ценности в истории была бесполезна.

Перейти с понятия ценности (value) на влияние (impact). Ваша история не обязательна должна иметь ценность, но обязательно должна оказывать влияние на того актера, что указан в истории. А уже это влияние ведет в конечном итоге к цели, которая имеет для вас ценность.

Пример:

Представим что вы создали историю — «Как инвестиционный аналитик я получаю отчет №17 об инвестициях чтобы БЫСТРЕЕ принять решение».

У меня Acceptance Criteria — это метрика на value в US. Как померить такой value? Как понять что аналитик принял решение быстрее? Как вы поймете в конце что история выполнена?

Переделаем историю на влияние — «Как инвестиционный аналитик я получаю отчет No17 об инвестициях БЫСТРЕЕ». То есть сейчас этот отчет формируется за 60 сек. Вы указываете в АС что отчет должен формироваться за 15 сек. В конце понятно выполнено ли АС, понятно какие влияние вы оказали на работу аналитика.

Но в чем ценность того, что аналитик стал получать отчет быстрее?

Здесь можно перейти к общей постановке Цели для продукта. Чтобы прийти к такой истории вы:

1. Вы построили Impact mapping.

2. Вы определили Цель и метрику на нее. Например, «ускорение сроков согласования инвестиционных бюджетов».

3. Вы определили что «инвестиционный аналитик» может вам помочь в достижении этой цели.

4. Вы сделали предположение что если аналитик будет получить отчет №17 быстрее, то это приведет вам к вашей цели.

5. Потому данная история — это проверка данного предположения достижения цели. То есть смысл Impact map — это трассировка от User story к общей Цели продукта. Если такой связи нет и вы не можете ее найти — значит вы делаете что-то бесполезное.

ДРУГИЕ ПРИМЕРЫ USER STORY

Наиболее распространенные ошибки при написании историй

История для юзера

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

На первый взгляд, вы не увидите в этой истории никаких изъянов – все элементы на месте. А теперь расскажите-ка, для кого вы собираетесь сделать эту фичу и что этот юзер знает об управлении объявлениями? Он администратор портала объявлений, которому нужно время от времени чистить базу и премодерировать объявления? Или, может, он рекламодатель, которому нужно просматривать список созданных им объявлений и иметь возможность удалять ненужные объявления прямо из этого списка?

Вы могли заметить, что у этих двух пользователей совсем разные роли, с разными ожиданиями с разными требованиями к системе. Основная ошибка этой истории – игнорирование роли и персоны пользователя.

Никакой бизнес ценности для пользователя

Пример:«Как рекламодатель, я хочу чтобы у меня была возможность фильтровать объявления»

У нас есть роль, есть потребность, но причина или бизнес ценность куда-то запропастились. Зачем рекламодателю фильтровать объявления? Чего он хочет достигнуть? Не думайте, что это буквоедство, история действительно теряет смысл без нужных элементов.

Никаких критериев приемки

В любом из примеров приведенных выше, нет критериев приемки. Истории могут проваливать тесты, или тест кейсы могут проверять не те критерии из-за отсутствия понимания того, как должен выглядеть конечный результат и каким требованиям он должен соответствовать. Критерии приемки нужны именно для того, чтобы рассеять ложные предположения, а иногда даже перепланировать историю или разбить ее на меньшие.

Практические советы по написанию пользовательских историй

• Лучше написать много историй поменьше, чем несколько громоздких.

• Каждая история в идеале должна быть написана избегая технического жаргона – чтобы клиент мог приоритезировать истории и включать их в итерации.

• *Истории должны быть написаны таким образом, чтобы их можно было протестировать*

• Тесты должны быть написаны до кода.

• Как можно дольше стоит избегать UI. История должна выполняться без привязки к конкретным элементам.

• Каждая история должна содержать оценку.

• История должна иметь концовку – т. е. приводить к конкретному результату.

• История должна вмещаться в итерацию.

После написания User Story их необходимо Детализировать и Сгрупировать и Упорядочить.

Диаграмму вариантов использования есть смысл строить во время изучения технического задания, она состоит из графической диаграммы, описывающей действующие лица и прецеденты, а также спецификации, представляющего собой текстовое описание конкретных последовательностей действий (потока событий), которые выполняет пользователь при работе с системой. Кроме того, use-case диаграмма достаточно проста, чтобы ее мог понять заказчик, следовательно вы можете использовать ее для согласования ТЗ (ведь диаграмма описывает функциональные требования к системе).

На диаграмме использования изображаются:

• акторы – группы лиц или систем, взаимодействующих с нашей системой;

• варианты использования (прецеденты) – сервисы, которые наша система предоставляет акторам;

• комментарии;

• отношения между элементами диаграммы.

На наш взгляд, наиболее правильный порядок построения диаграммы следующий:

1. выделить группы действующих лиц (работающих с системой по-разному, часто из-за различных прав доступа);

2. идентифицировать как можно больше вариантов использования (процессов, которые могут выполнять пользователи). Идентификация вариантов использования должна осуществлятся на основе описанных ранее Пользовательских историй (User story). При этом не следует делить процессы слишком мелко, нужно выбирать лишь те, которые дадут пользователю значимый результат. Например, кассир может «продать товар» (это будет являться прецедентом), однако «ввод штрих-кода товара для получения цены» самостоятельным прецедентом не является;

3. дополнить прецеденты словесным описанием (сценарием):

• для каждого прецедента создать разделы: «главная последовательность» и «альтернативные последовательности»;

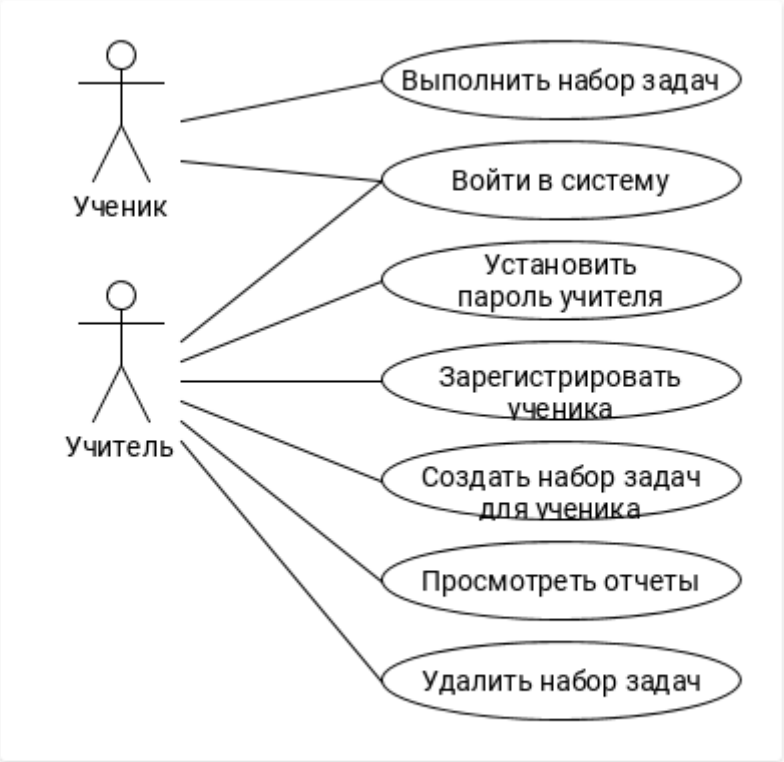
• при составлении сценария нужно упорно задавать заказчику вопросы «что происходит?», «что дальше?», «что еще может происходить?» и записывать ответы на них.

Сценарии являются очень важной частью диаграмм использования, хотя их формат и не регламентирован. Ряд авторов предлагает использовать *псевдокод* для представления сценария и даже сразу строить *диаграммы деятельности* или *взаимодействия*, но на наш взгляд, наиболее предпочтительным вариантом на этапе построения use-case диаграмм является текстовый, описывающий систему с точки зрения пользователя (т.к. именно этот формат будет наиболее понятен заказчику, с которым вам предстоит *согласовывать техническое задание*).

Рассмотрим разработку диаграмм вариантов использования на примере – пусть заказчик дал нам следующее техническое задание:

> *Цель – развитие у детей математических навыков.*
> *Платформа: Linux, Windows, Android.*
> *Функциональность:*
>
> - *для учеников:*
>   - *выбор подготовленного учителем блока заданий;*
>   - *выполнение заданий;*
> - *для учителя:*
>   - *подготовка для учеников блоков заданий;*
>   - *добавление в систему ученика;*
>   - *просмотр отчетов.*
>
> *При первом запуске система должна позволять ввести пароль учителя. Задания представляют собой математические задачи на сложение, вычитание, умножение и деление. В блоке задач могут быть задачи различных типов (указывается количество). Помимо ввода типа выполняемой в примере операции необходимо указывать допустимые диапазоны чисел (или даже отдельные числа, т.к. при изучении таблицы умножения часто сначала учат умножение на 2, затем на 5, а только потом все остальное). Кроме того, для операции вычитания необходимо иметь возможность установить вычитаемое меньше уменьшаемого (т.к. в противном случае результат будет отрицательным, а отрицательные числа в школе проходят гораздо позже).*

Очевидно, несмотря на то, что заказчик очень подробно описал некоторые детали, мы не можем не только приступить к реализации задачи, но даже приблизительно оценить стоимость и сроки выполнения. Из такого задания не понятно, например, что должны содержать отчеты. Однако, мы сразу можем выделить две группы пользователей и несколько видов их деятельности.



Пример диаграммы использования

Сплошные линии на диаграмме представляют собой отношения ассоциации, отражающие возможность использования актором прецедента. После того, как определен набор вариантов использования, можно приступать к составлению сценариев. Сценарии должны описываться с точки зрения пользователя, при этом важно описывать взаимодействие пользователя с элементами интерфейса. Так например сценарий прецедента регистрации ученика мог бы выглядеть следующим образом:

***Название прецедента**: регистрация ученика*
***Действующее лицо**: учитель*
***Цель**: добавить ученика в систему, получив его пароль*
***Предусловия**: учитель осуществил вход в систему*
***Главная последовательность**:*
1. учитель выбирает в главном меню пункт «добавить ученика«;
2. система показывает учителю окно добавления ученика, содержащее поля для ввода логина и пароля, а также кнопки «далее» и «назад»;
3. учитель вводит желаемый логин и пароль ученика, нажимает кнопку «далее»;
4. система добавляет ученика;
5. учителю открывается главное меню и в течении 5 секунд выводится уведомление о том, что ученик был добавлен успешно.
***Альтернативная последовательность** (возврат в главное меню без добавления ученика):*
3.1.а. учитель нажимает кнопку «назад»;
3.1.б. учителю открывается главное меню (при этом данные, введенные в формы окна добавления ученика не сохраняются).
***Альтернативная последовательность** (добавление ученика, уже имеющегося в системе):*
3.2.а. учителю в течении 5 секунд отображается уведомление о том, что запрашиваемый логин занят.
!!!ПРИМЕРЫ ОПИСАНИЯ USE_CASE (ПРЕЦЕДЕНТОВ)!!!

Аналогичным образом должны быть прописаны все прецеденты, изображенные на диаграмме.

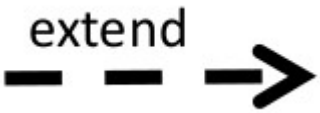
Несмотря на простоту приведенного сценария, в его последовательностях можно найти дублирование, если оно имеет место в ваших сценариях – вы можете выделить некоторые фрагменты описания в отдельные прецеденты (которые могут быть как самостоятельными, так и являться лишь частью других вариантов использования). При этом между прецедентами появится **отношения**

• ***Простая ассоциация*** - отражается линией между актером и вариантом использования (без стрелки). Отражает связь актера и варианта использования.На рисунке между актером *администратор* и вариантом использования просматривать *заказ*.

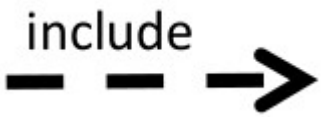
• ***Направленная ассоциация*** - то же что и простая ассоциация, но показывает, что вариант использования инициализируется актером. Обозначается стрелкой.

• ***Наследование*** - показывает, что потомок наследует атрибуты и поведение своего прямого предка. Может применяться как для актеров, так для вариантов использования.

• ***Расширение*** (extend) - показывает, что вариант использования расширяет базовую последовательность действий и вставляет extend собственную последовательность. При этом в отличие от типа отношений "включение" расширенная последовательность может осуществляться в зависимости от определенных условий.



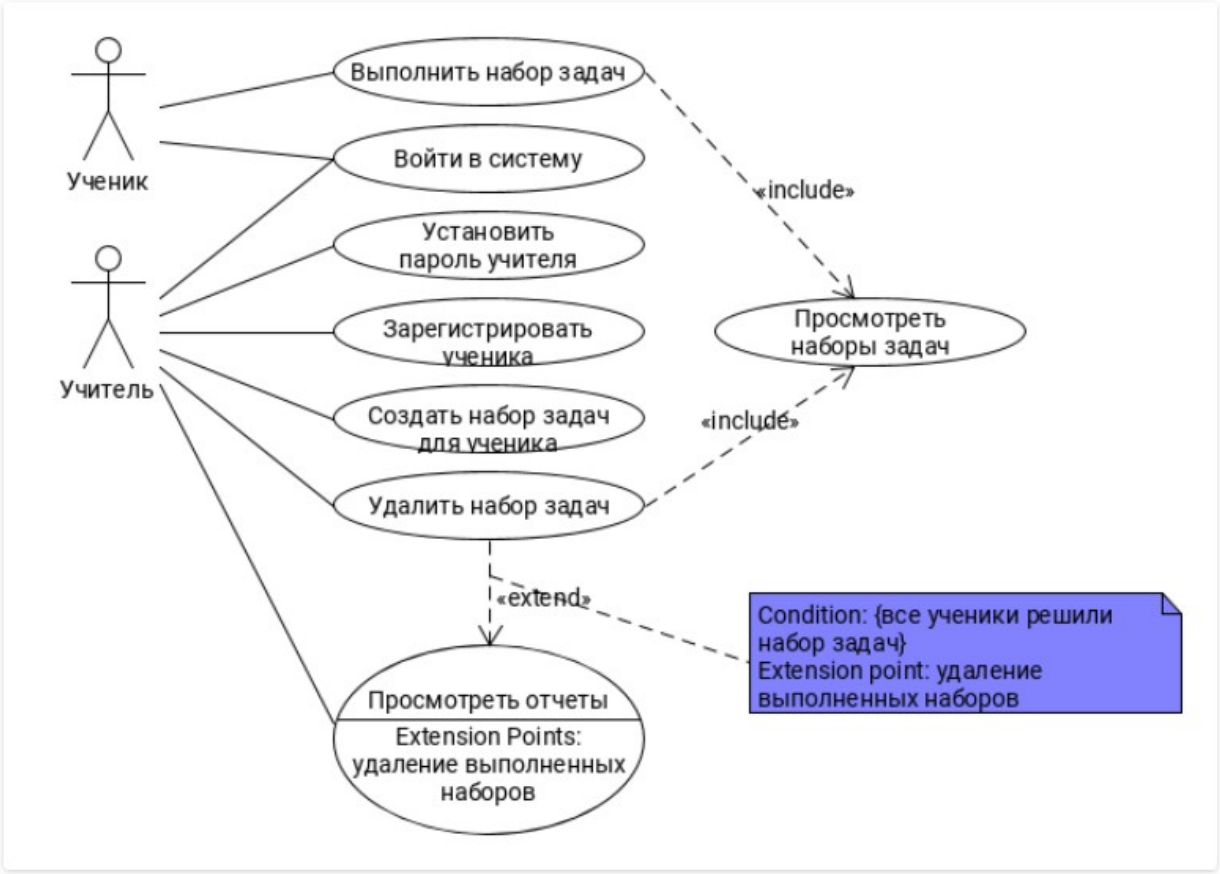
• ***Включение*** (include)- показывает, что вариант использования include включается в базовую последовательность и выполняется всегда.



Отношение включения указывает на то, что поведение одного прецедента включается в некоторой точке в другой прецедент в качестве составного компонента. Особенности включения заключаются в том, что включаемый прецедент должен быть обязательным для дополняемого (включение должно быть безусловным, а дополняемый вариант использования без включения не сможет выполняться), т.е. это отношение задает очень сильную связь. Например, если мы хотим изобразить на диаграмме тот факт, что удаление набора задач учителем и выполнение задач учеником не должно происходить без *обязательного* просмотра всех наборов задач – то нам нужно использовать отношение включения:

Отношение включения на диаграмме использования

Отношение расширения отражает *возможное* присоединение одного варианта использования к другому в некоторой точке (*точке расширения*). При этом подчеркивается то, что расширяющий вариант использования выполняется лишь при определенных условиях и не является обязательным для выполнения основного прецедента. На диаграмме такой вид отношения изображается стрелкой, направленной к расширяемому прецеденту, в отдельном разделе которого *может быть описана* точка расширения, а условия расширения *могут быть приведены* в комментарии с *ключевым словом* Condition. Таким образом, расширение позволяет моделировать *необязательное поведение системы*, которое является условным и *не изменяет поведение* основного прецедента. Например отношение расширения нужно применить, если по техническому заданию *требуется возможность* удаления набора задач в *прецеденте просмотра отчетов* при условии, что все ученики решили этот набор.



Отношение расширения на диаграмме использования

Таким образом можно показать, что у учителя появляется возможность (но не обязанность) удалить набор задач при просмотре отчетов если все ученики выполнили этот набор.

Инструментарий: creately.com/, draw.io или MS Visio

Пример разработки User Story

Как водитель с загоревшейся лампочкой бензина я хочу быстро найти ближайшую хорошую заправку, чтобы заправиться качественным бензином.

Критерии приемки:

1. US01. Как водитель с загоревшейся лампочкой я могу просмотреть все ближайшие заправки, чтобы успеть заправиться до полного расхода топлива.

2. US02. Как ... я могу выбрать заправки подходящих мне брендов АЗС, для использования карт лояльности.

3. US03. Как ... я могу видеть ближайшие заправки выбраненных брендов списком.

4. US04. Как ... я могу видеть ближайшие заправки выбранных на карте.

Обработка ошибок:

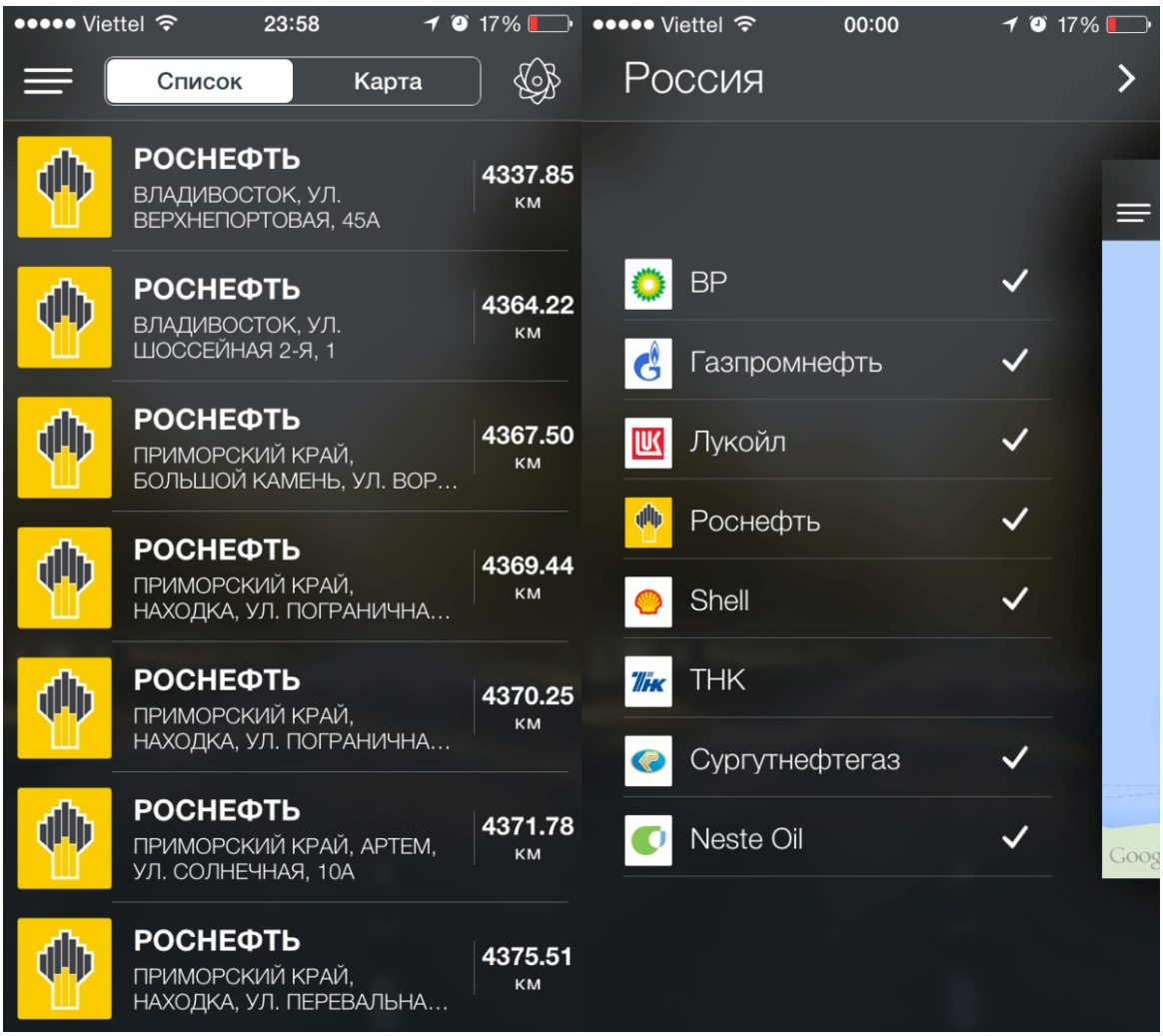
1. При выключенной геолокации пользователя необходимо дать ему информацию о том, где ее включить.

Технические заметки:

1. Заправки в списке должны обновляться при изменении местоположения пользователя на 100 метров.

Такая формулировка задачи помогает разработчикам и тестировщикам не пытаться сравнивать готовую задачу с ui требованиями, которые быстро устаревают, а смотреть на основные проблемы, которые должны быть решены в рамках задачи. Дополненная прототипами, такая история легко становится задачкой в джире или бейскемпе, которую можно делать даже без финального дизайна.

Вот как выглядели экраны, относящиеся к этой истории, в итоговом приложении:



Итак, истории:

1. Как пользователь я могу хранить свои фотографии в системе, чтобы иметь возможность показать или продать их другим пользователям.

2. Как рекламодатель я могу помещать свою рекламу в системе, ориентированную на пользователей.

3. Как администратор я могу управлять фотографиями пользователей, так чтобы контент сайта был легальным.

Во время обсуждения первой истории, заказчик и команда приходят к тому, что пользователи системы должны быть авторизированны системой перед выполнением каких-либо действий с фотографиями. Это приводит к появлению новой пользовательской роли «гостя» – группе людей, которые неавторизированны системой или вообще пока не имеют пользовательской учетной записи.

4 Как гость я могу зарегистрироваться в системе для получения пользовательской учетной записи и последующей работы.

5 Как гость я могу войти в систему под ранее созданной учетной записью, для последующей работы.

Пользуясь принципом симметричности требований, команда и заказчик принимают решение, что пользователь должен иметь возможность удалить свою учетную запись в случае необходимости:

6 Как пользователь я могу удалить свою учетную запись и перестать быть пользователем системы.

Обсуждая концепцию учетных записей, рождаются также следующие истории:

7 Как пользователь я могу изменить данные своей учетной записи.

8 Как пользователь я могу сделать некоторые поля своей учетной записи видимыми для других пользователей.

ПРИМЕР ДЕТАЛИЗАЦИИ.

Рассмотрим одну из историй, идентифицированную выше:

4 Как гость я могу зарегистрироваться в системе для получения пользовательской учетной записи и последующей работы.

Во время обсуждения этой истории с командой заказчику задают вопрос о том какая информация нужна для создания пользовательской учетной записи. Обсуждая различные варианты, заказчик и команда приходят к тому, что для первой версии системы достаточно будет проверенного электронного адреса плюс имени пользователя и его пароля.

К истории дописывается этой комментарий. Теперь история выглядит так:

4 Как гость я могу зарегистрироваться в системе для получения пользовательской учетной записи и последующей работы.

Нужен проверенный email и выбранные пользователем имя и пароль.

В ходе дальнейших высказываний кто-то из тестировщиков задает резонный вопрос о минимальной длине пароля и проверке на уникальности имени. Продолжая дискуссию, команда и заказчики приходят к мнению, что необходимо описать основные критерии готовности истории, чтобы команда понимала ожидания и знала, когда объявлять историю готовой:

4 Как гость я могу зарегистрироваться в системе для получения пользовательской учетной записи и последующей работы.

Нужен проверенный email и выбранные пользователем имя и пароль.

Тест 1: пользователь не может ввести пароль меньше 6 символов.

Тест 2: пользователь не может ввести имя меньше 3 и больше 20 символов.

Тест 3: пользователь должен иметь уникальное имя в системе.

Тест 4: после регистрации пользователь должен получить имейл для активизации своей учетной записи.

Тест 5: пользователь не может войти в систему, если учетная запись не была активизирована.

Тест 6: при успешном входе система приветствует пользователя текстом «Добро пожаловать, <имя пользователя>».

Возможно во время реализации, тестирования и приема истории возникнут ещё какие-то дополнительные моменты. В этом случае они могут быть описаны в виде уточняющих тестов или как комментарии. Возможно из этих дополнения появятся новые истории.

Таким образом истории пополняются деталями по мере необходимости, эволюционируя от коротких высказываний до детализированных и согласованных требований со встроенными критериями готовности.

МОЩНЫЕ ИНСТРУМЕНТЫ РАБОТЫ С ИСТОРИЯМИ: УПОРЯДОЧИВАНИЕ, РАЗБИЕНИЕ И ГРУППИРОВКА

Как видно, описанные выше истории являются более-менее автономными сущностями, и, как следствие, могут быть перечислены в другом порядке. Конечно между историями существуют связи и логические цепочки – нельзя, к примеру, удалять пользовательские записи, не умея создавать их. Но все таки можно научиться составлять истории таким образом, чтоб обеспечить некоторую свободу в выборе порядка их реализации. Свободы будет, естественно, тем больше, чем больше самих историй и чем независимее они друг от друга.

Если же истории независимы, да к тому же их достаточно много, то можно смело предположить, что их ценность с точки зрения вклада в систему различна. А значит, варьируя

порядком историй, можно выставить их в таком порядке, что первые «n» историй будут играть ключевую роль в полезности системы, в то время как другие истории будут скорее необязательными добавками, привлекающими пользователей или облегчающими их работу.

Пользуясь знанием рынка, а также здравым смыслом (к сожалению на сегодняшний день оба этих критерия не поддаются численной оценке), заказчик выстраивает список историй таким образом, чтобы максимизировать возврат вложений от проекта.

Вот пример, как могли бы быть отсортированы истории вышеописанного проекта (это всего лишь один из вариантов, конечно, есть и другие):

4 Как гость я могу зарегистрироваться в системе для получения пользовательской учётной записи и последующей работы.

5 Как гость я могу войти в систему, имперсонализируясь с ранее созданной учётной записью, для последующей работы.

1 Как пользователь я могу хранить свои фотографии в системе, чтобы иметь возможность показать или продать их другим пользователям.

3 Как администратор я могу управлять фотографиями пользователей, так чтобы контент сайта был легальным.

7 Как пользователь я могу изменить данные своей учетной записи для корректировки измененных или неверных данных.

2 Как рекламодатель я могу помещать свою рекламу в системе, ориентированную на пользователей.

8 Как пользователь я могу сделать некоторые поля своей учетной записи видимыми для других пользователей.

6 Как пользователь я могу удалить свою учетную запись и перестать быть пользователем системы.

Как вы ведите, истории выстроены в порядке, который, во-первых, логичен с точки зрения заказчика и команды, а во-вторых ценность историй уменьшается сверху вниз. Таким образом, если, к примеру, на половине проекта наступает нехватка ресурсов (скажем, после реализации истории для администратора системы), заказчики смогут получить выгоду от продукта, так как наиболее важные истории уже будут реализованы. Это ни что иное как минимизация рисков от вложений.

Конечно, порой не так легко и очевидно принять правильное решение о порядке историй, но в этом и состоит мастерство быть заказчиком (это отдельная, неисчерпаемая тема…).

Кроме инструментария ранжирования историй, в руках у заказчика есть и другие мощные средства, позволяющие повысить эффективность своих финансовых вложений. К примеру, одна из описанных на ранней фазе проекта историй в какой-то момент может показаться слишком большой в сравнении с другими, что усложняет понимание её приоритета:

1 Как пользователь я могу хранить свои фотографии в системе, чтобы иметь возможность показать или продать их другим пользователям.

В этом случае заказчик и команда могут попробовать разбить ее на несколько более мелких историй, каждая из которых может получить свой приоритет:

9 Как пользователь я могу хранить свои фотографии в системе, чтобы иметь возможность показать их другим пользователям.

10 Как пользователь я могу хранить свои фотографии в системе, чтобы иметь возможность продать их другим пользователям.

При этом нужно учесть, что начальная история не разбивается на две «под-истории», а замещается двумя новыми. Это не разбиение историй на подзадачи для постановки их программистам, это всего лишь переформулировка требований для более эффективного управления ими.

Подобный процесс разбиения сложных и больших истории на более простые может осуществляться в теории довольно долго. На практике же, заказчики и команда в скором времени вырабатывают совместное понимание адекватного размера историй и следуют ему при написании новых и разбиении существующих историй. Этот размер зависит от количе-

ства историй, реализуемых за итерацию. Но об этом поговорим подробнее, обсуждая планирование.

Механизмом, обратным разбиению, служит группировка историй. Иногда бывает полезно склеить мелкие истории в одну побольше для улучшения понимания связности историй.

UC.US.02 Управление комментариями к объявлениям

Актёр: пользователь
Триггер: желание пользователя написать комментарий
Предусловие: пользователь авторизован и у него на экране открыто объявление

Основной сценарий 1 (написание комментария):

1. Пользователь вводит текст в поле для комментария
2. Пользователь подтверждает отправку комментария.
3. Система сохраняет комментарий.
4. Система отображает комментарий к объявлению.

Альтернативные сценарии:

3.а. Системный сбой, сбой соединения: Система не сохраняет комментарий.
  3.а.1. Система возвращается к шагу 1 основного сценария 1.

Основной сценарий 2 (редактирование комментария)

1. Пользователь инициирует вызов режима редактирования комментария.
2. Система отображает режим редактирования.
3. Пользователь меняет значение поля с комментарием.
4. Пользователь подтверждает сохранение комментария.
5. Система сохраняет изменённый комментарий.
6. Система отображает измененный комментарий в объявлении.

Альтернативные сценарии:

5.а. Система отображает предупреждение о том, что поле не может быть пустым.
  5.а.1. Система возвращается к шагу 1 основного сценария 2.
5.б. Система не сохраняет изменённый комментарий.
  5.б.1. Система возвращается к шагу 1 основного сценария 2.

Основной сценарий 3 (удаление комментария):

1. Пользователь инициирует удаление своего комментария.
2. Система запрашивает у пользователя подтверждение на удаление комментария.
3. Пользователь подтверждает удаление комментария.
4. Система удаляет комментарий из базы данных.
5. Система помещает вместо соответствующего комментария уведомление о его удалении.

Альтернативные сценарии:

3.а. Пользователь отменяет удаление комментария.
  3.а.1. Система возвращается к шагу 1 основного сценария 3.

UC.US.03 Обмен личными сообщениями

Актёр: пользователь
Триггер: необходимость отправить/просмотреть сообщение
Предусловие: пользователь авторизован в системе

Основной сценарий 1 (просмотр и ответ на сообщение):

Предусловие: пользователь находится в личном кабинете

1. Пользователь выбирает раздел «Мои сообщения» в личном кабинете.
2. Система отображает список диалогов.
3. Пользователь выбирает диалог.
4. Система открывает диалог.
5. Система отображает поле для ввода сообщения.
6. Пользователь вводит текст.
7. Пользователь выбирает «Отправить».
8. Система сохраняет сообщение.
9. Система отображает новое сообщение в диалоге.

Альтернативные сценарии:

9.а. Система не сохраняет сообщение.
9.а.1. Система возвращается к шагу 6 основного сценария 1.

Основной сценарий 2 (отправка сообщения пользователю):

Предусловие: открыто объявление пользователя, которому необходимо написать личное сообщение

1. Пользователь выбирает в окне объявления «Отправить сообщение».
2. Система отображает поле для ввода текста.
3. Пользователь вводит текст.
4. Пользователь выбирает «Отправить».
5. Система сохраняет сообщение.
6. Система возвращается к шагу 1 основного сценария 2.

Альтернативные сценарии:

5.а. Система выдаёт сообщение о том, что «Сообщение не может быть отправлено».
5.а.1. Система возвращается к шагу 1 основного сценария 2.
4.а. Пользователь выбирает «Отмена».
4.а.1. Система возвращается к шагу 1 основного сценария 2.