# Практическая работа №4

**Разработка пользовательских историй при проектировании ИС**

Цель работы: освоение технологии разработки и управления требованиями

Задание: Ознакомиться с теоретическим материалом, разработать не менее 16 пользовательских историй от различных типов пользователей

Теоретический материал:

В практике подготовки требований, которые в дальнейшем будут составлять каркас разрабатываемого программного продукта, принято использовать один из стандартизированных подходов для их описания - Use Cases ("пользовательские истории").

*Use Case ("пользовательская история", "юскейс" и т. д.) - это сценарная пошаговая техника описания взаимодействия двух или более участников, задействованных в автоматизации. С помощью Use Case может быть описано и пользовательское требование, и требование к взаимодействию систем, и описание взаимодействия людей и компаний в реальной жизни. В общем случае с помощью Use Case может описываться взаимодействие двух или большего количества участников, имеющее конкретную цель.*

В разработке программного обеспечения этот способ разработки и управления требованиями часто применяют для проектирования и описания взаимодействия пользователя и системы, поэтому название Use *Case* часто воспринимают как *синоним* требования человека-пользователя к решению определенной задачи в системе.

Пользовательские истории формулируются как одно или более предложений на "повседневном" языке пользователя. Они получаются относительно небольшие *по* объему, что удобно как для составления, так и для их обсуждения, приоритизации, планирования, оценки и дальнейшей работы с ними. Пользовательские истории получаются в виде алгоритма действий пользователя с реализуемым программным продуктом. Адекватная оценка трудоемкости историй позволяет планировать сроки ее реализации, тем самым управляя содержимым спринтов. Разработать оптимальную пользовательскую историю не так просто, нужен определенный навык, который позволит создать качественное описание требований пользователей к реализуемому программному продукту, но в награду за это получаются следующие преимущества:

* *Краткость.* Пользовательская история описывает небольшую часть бизнес-ценности, которую возможно реализовать за период спринта.
* *"Незатратность" создания и сопровождения.* За счет своей "компактности" пользовательские требования достаточно просто создать и сопровождать их изменения на всем протяжении жизненного цикла продукта.
* *Вовлечение ключевых пользователей в процесс создания продукта.* За счет своей доступности бизнес-требования смогут стать реальным "мостиком" между пользователями и командой разработчиков. Это позволит более адекватно управлять ожиданиями пользователей и вовлечь их на нужную степень погружения в процесс разработки продукта.
* *Облегчают оценку заданий.* Формат пользовательских историй способствует более точной оценке необходимых системных разработок/доработок.

Чем более компактный объем имеет пользовательская история, тем проще выполнять ее оценку. Это приводит к более верным оценкам сроков реализации программного продукта и к более понятному планированию выполняемых *работ*. При этом важно знать меру в уменьшении и дроблении Use cases. Если сделать их слишком много, то у вас получится огромный *список* задач в бэклоге продукта, и *процесс управления* ими и фиксации на карте историй усложнится.

Для пользователей требования, выраженные в виде пользовательских историй, являются основным инструментом влияния на разрабатываемый *программный продукт*. Пользовательские истории определяют формат, в котором у пользователей есть возможность отразить все те важные факторы, которые, *по* их мнению, должны быть учтены в процессе автоматизации.

Важно отдавать себе отчет в том, что пользовательские истории не обеспечивают полноту всех функциональных требований и имеют ряд недостатков:

* *Не масштабируются для больших программных продуктов.* Пользовательские истории хорошо себя зарекомендовали, когда речь идет о создании небольших или средних по объемам и сложности программным продуктам. За счет того, что этот вид работы с требованиями ориентирован на непосредственную работу с пользователями и поддерживается "бизнес-лексиконом", он неприменим в работе над крупными информационными системами, когда на первый план выходит организационная структура проекта/процесса, и важны формальные признаки сдачи/приемки работ.
* *Требовательны к квалификации разработчиков.* Разработчики, работающие с пользовательскими историями, должны обладать высокой квалификацией и неплохими коммуникативными навыками, которые позволят им получить необходимые уточнения уже изложенных требований. Как правило, таких разработчиков немного. И это еще один неоспоримый недостаток.
* *Не являются средством документирования.* Пользовательские истории - это небольшое и удобное представление информации. Они сформулированы на ежедневном языке пользователя и содержат небольшие детали, оставаясь открытыми для интерпретации. Они помогают понимать, что должна делать система, но при этом пользовательских требований недостаточно, чтобы понять, как будет организована логика системы. Пользовательские требования являются необходимой "верхушкой" для понимания назначения информационной системы, но для реализации системы разработчику приходится додумывать множество значимых деталей.

Инженерия требований - важный этап в создании каждого программного продукта. Функциональные обязанности *по* извлечению, разработке и управлению требованиями ложатся на всех членов команды, именно поэтому в процессе работы с ними задействованы такие артефакты, как пользовательские истории, назначение которых - облегчить получение информации, необходимой для качественной реализации программного продукта.

Формат пользовательской истории как правило следующий:

Я как *тип пользователя*, хочу *действие*, для того чтобы *цель.*

Пример:

Я как покупатель, хочу сформировать отчет по покупкам, для того чтобы пороверить свои покупки

Придерживаться такой структуры необязательно, но она помогает определить критерии готовности работы. История выполнена, когда упомянутый тип пользователя получает требуемую ценность. В идеале, команды формулируют свою собственную структуру и придерживаются ее.