Технический долг. Устранение долга

При разработке программ иногда приходится жертвовать качеством кода ради скорости разработки. Чтобы не нарушать намеченный дедлайн, зачастую приходится использовать некоторые «костыли» и «хаки». Их исправление обычно переносят на будущие релизы – что, собственно, и называют «техническим долгом».

На протяжении 2-х спринтов в силу сжатых сроков и высокой нагруженности разработчиков “Технический долг” на нашем проекте накапливался. В коде проекта можно было встретить *повторяющийся код*, в редких случаях даже *нечитабельность*. Но самой большой проблемой в нашем проекте на тот момент являлось неправильное использование контейнера состояний Redux. Это приводило к запутанной *архитектуре приложения*. Новые фичи требовали больше времени на внедрение и приводили к созданию *ненужных и сложных зависимостей,* что в свою очередь к накапливанию “технического долга”. Наша кривая “технического долга” на тот момент выглядела так:

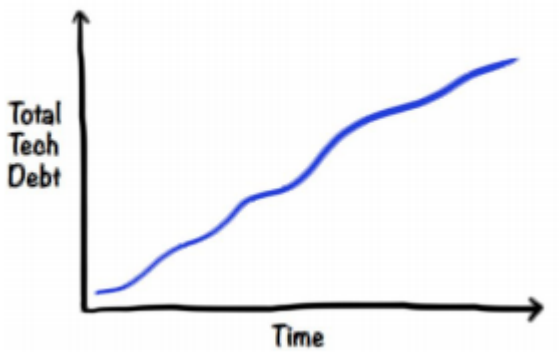

Рис. 1 – Кривая “технического долга” на момент 2-го спринта.

Оценки качества кода проекта на 2-ой спринт от членов команды:

- Игорь Лосев – 3 / 5
- Герман Волков – 2 / 5
- Никита Павец – 3 / 5
- Коля Быльнов – 3 / 5

Посовещавшись всей командой, мы пришли к выводу, что нашему проекту нужен рефакторинг. Мы решили, что сделав это мы проиграем в краткосрочной перспективе, но выиграем в долгосрочной. План мероприятий с датами проведения выглядел так:

- 21.04.2017 – 29.04.2017: Полный рефакторинг архитектуры нашего приложения. Корректное использование контейнера состояний Redux

- 30.04.2017 – 01.04.2017: Рефакторинг, исключающий использование повторяющегося и нечитабельного кода, устранение “костылей”.

Несмотря на большой объем рефакторинга и задач на спринт, нам удалось уложиться в поставленные сроки. После проведения рефакторинга и, соответственно, на данный момент наша кривая “технического долга” стала выглядеть следующим образом:

Рис. 2 – Кривая “технического долга” на момент 3-го спринта.