RuStore

Емельянов Михаил

Как мы победили техдолг в проекте RuStore

Mobius 2024

Содержание

01 Проблематика

02 Анализ и оценка

03 Аудит

04 Стратегия изменений

05 Как исправляли

06 Результат

07 Итоги

08 Q&A

Емельянов Михаил RuStore

Кратко о себе и компании

О себе

- 15 лет в IT
- 12 лет в мобильной разработке
- Возглавляю департамент разработки RuStore
- Активный участник конференций
- Дополнительно занимаюсь R&D проектами с использованием AI

Что такое RuStore

Витрина для пользователей

Охват аудитории

Доступные приложения

Один из **основных Android сторов**

Консоль для разработчиков

Лучшие технологии и возможности

>50 М

Установок RuStore на устройства

40 000

Приложений доступно в RuStore

>10 000

Разработчиков — юридические и физические лица

Техдолг Legacy

Проблематика

Что такое техдолг?

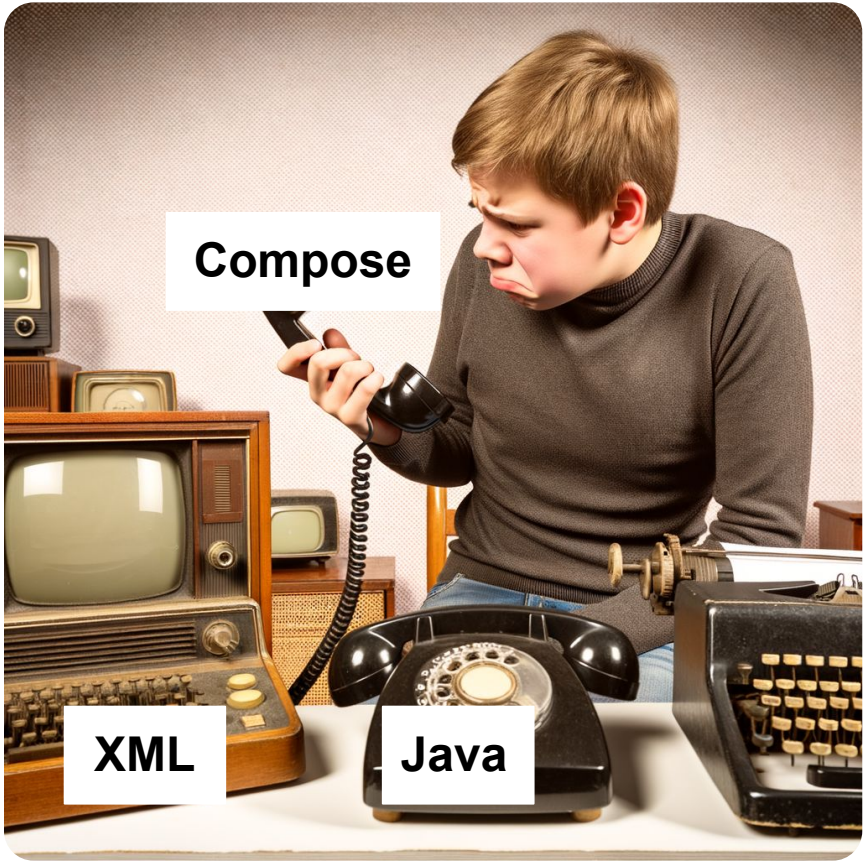
Что такое техдолг?

Это незавершенная задача
или принятые слабые технические
решения, в пользу ускорения
разработки для попадания в нужные
сроки релиза продукта

Примеры

Устаревшие технологии

- Нет времени изучать
- Нет опыта использования
- Страшно :)



Примеры

Нет Unit-тестов

- Зачем?
- Это долго
- Напишем потом

Примеры

Спагетти-код

- Сложно структурирован
- Сильные зависимости
- «Почему нельзя просто взять и сделать?»
  (с) Любой продакт

Некоторые
факты из индустрии

NOKIA 2000

Nokia столкнулась с трудностями из-за сложности и устаревания платформы Symbian, которую было **трудно обновлять и поддерживать**. Это привело к потере рыночной доли перед лицом конкуренции со стороны таких систем, как iOS и Android.

HealthCare.gov 2013

При запуске веб-приложения для страхования здоровья были серьёзные проблемы с производительностью и доступностью. **Причиной стал переусложнённый и недостаточно протестированный код**. Стоимость проекта составила около 1,7 миллиарда долларов, и значительная часть этих средств ушла на исправление недочетов после запуска.

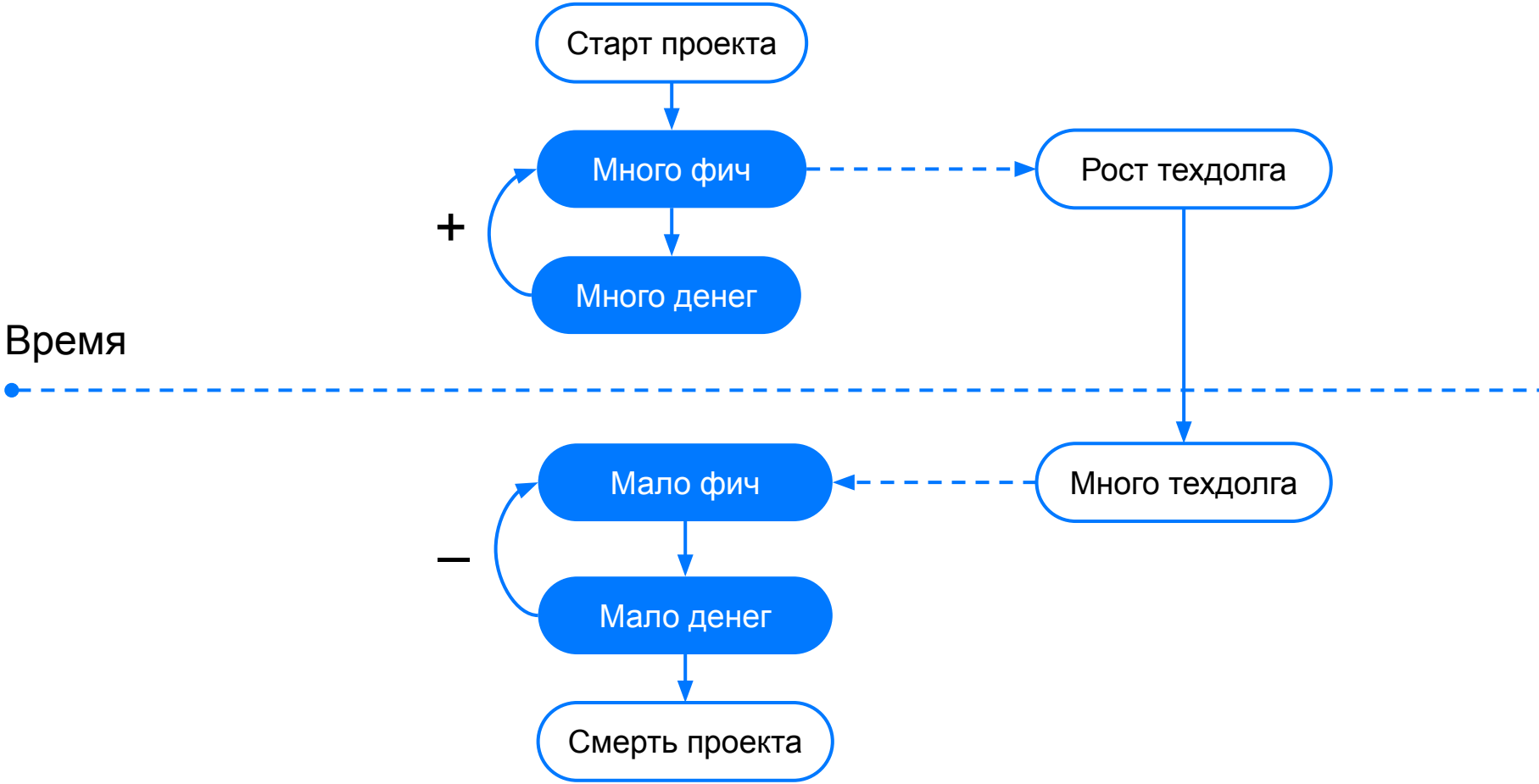
Knight 2012

В результате ошибки в коде торговой системы компания потеряла 440 миллионов долларов за несколько минут. Проблема была связана с **плохо протестированным программным обеспечением**, что является примером технического долга, который привел к финансовым потерям.

DELTA 2016

Компания Delta Air Lines в августе 2016 году столкнулась с **непредвиденным сбоем казалось бы надежных систем бронирования**, регистрации на рейс и посадки, что вызвало простой в течение нескольких часов и отмену более 2000 рейсов.

Рост техдолга
приводит к смерти
продукта


Старт проекта
Много фич
Рост техдолга
+
Много денег
Время
Мало фич
Много техдолга
–
Мало денег
Смерть проекта

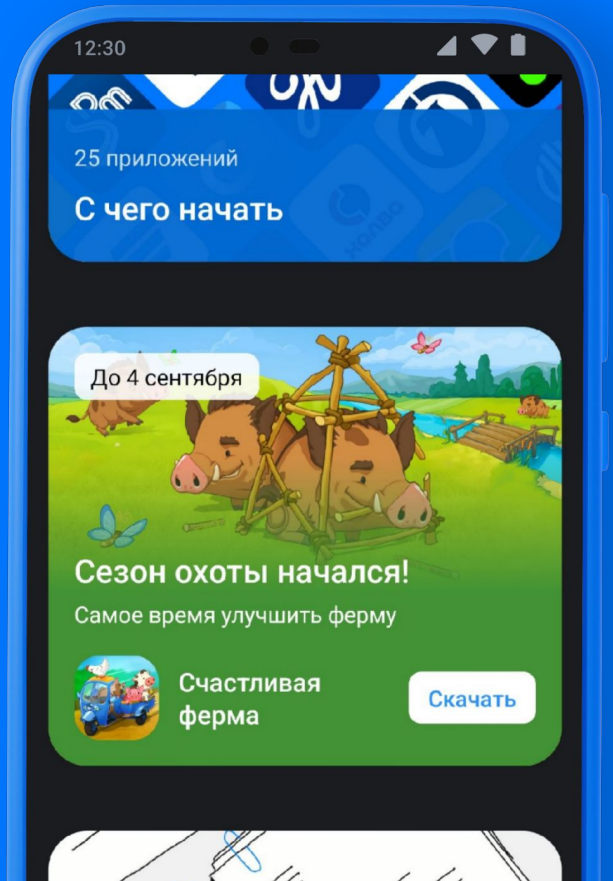
RuStore 2024

Наша ситуация

RuStore в 2022

Быстрый рост продукта и команды

- Запуск MVP за 3 месяца
- Добавление возможности монетизации через пол года
- Быстрый рост количества пользователей и разработчиков



Что ожидали в 2022-2023 гг

Метрики Методика

Анализ и оценка техдолга в RuStore

Сначала определиться —
что важно?

Метрики влияния техдолга

TTM	Stability	Scalable
Выпускать чаще	99.9%	Техдолг не растёт
Каждый релиз	Crash-Free	Легко добавлять, удалять код

Где и как искать?

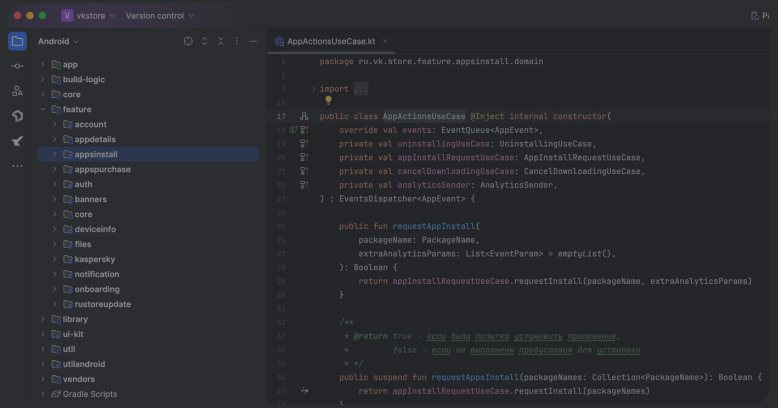
Где искать?

Domain слой — чаще всего масштабируется
Presentation — чаще изменяется

>50%
Кода проекта

Время сборки проекта влияет на TTM

- Проект
- Архитектура
- Код
- Паттерны
- Модули
- Слои
- Сборка



Аудит проекта

Собрать проблемы

Заводим проект-задачу аудита

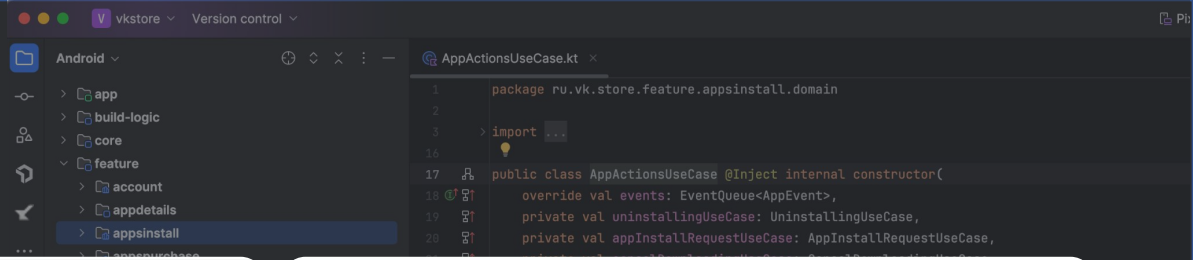
Находим технические проблемы,
заполняем по шаблону в задаче

~2-4 недели

Проблема

- Описание проблемы // *TODO*
- Важность (метрики) // *TODO*
- Скрины, примеры // *TODO*

Примеры



Бизнес-логика в экстеншенах

Описание:

Много примеров по коду, когда БЛ реализована в экстеншенах доменных моделей. Эту логику, как статику, можно протестировать. Но места, где она используется, невозможно протестировать изолированно.

Важность:

Влияет на качество unit-тестов.

Скрины, примеры:

if/else при подключении зависимостей в gradle

Описание:

Когда понадобилось сделать фичу по-быстрому, подключили таким образом вместо flavor.

Важность:

Будет негативно сказываться на кэшировании, скорости сборки.

Скрины, примеры:

Костыль с блокирующей навигацией

Описание:

Иногда есть потребность показать какой-то экран, например, экран форс-апдейта, а навигация на него может произойти асинхронно, и кто-то еще может попытаться показать что-то поверх этого экрана. Чтобы этого избежать, сделан костыль с блокирующей навигацией.

Важность:

Усложняет навигацию.

Скрины, примеры:

Лишние конфигурации Gradle

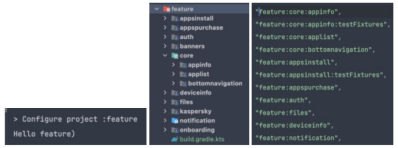
Описание:

Пути модулей указанный через ":" заставляют Gradle представлять директории в виде сабпроектов и потом постоянно конфигурировать

Важность:

Стреляет на скорости сборки на этапе конфигурации. Будет нагружать сборку по времени O (n^m), где n - количество сабпроджектов под фичи, а m - количество сабпроджектов представленных в виде директорий

Скрины, примеры:



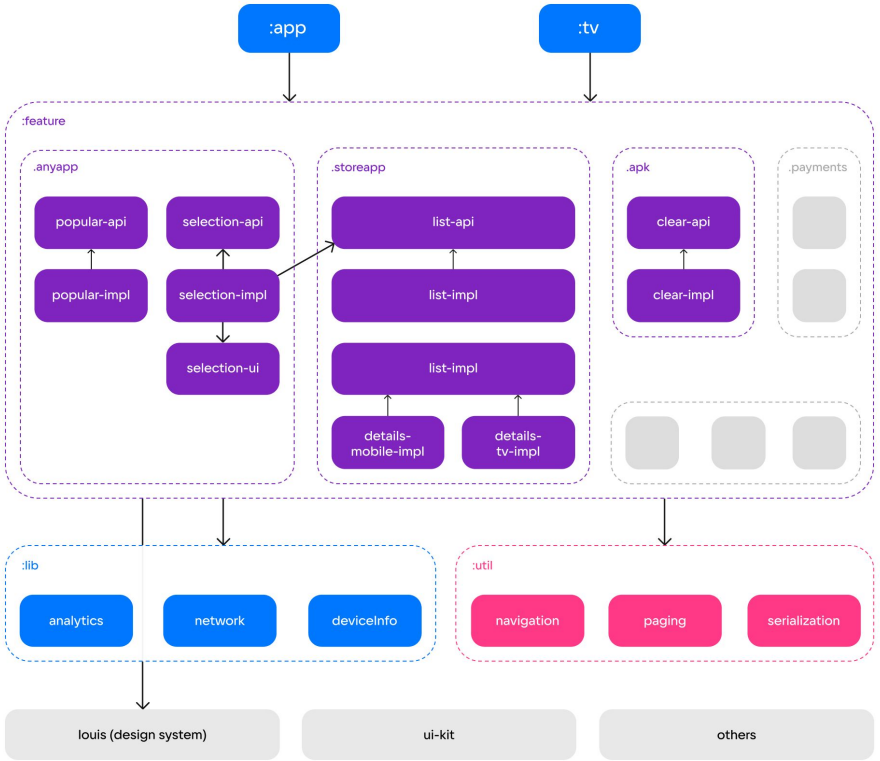
Архитектура Парадигмы

Стратегия изменений

Структура проекта

TTM Scalability

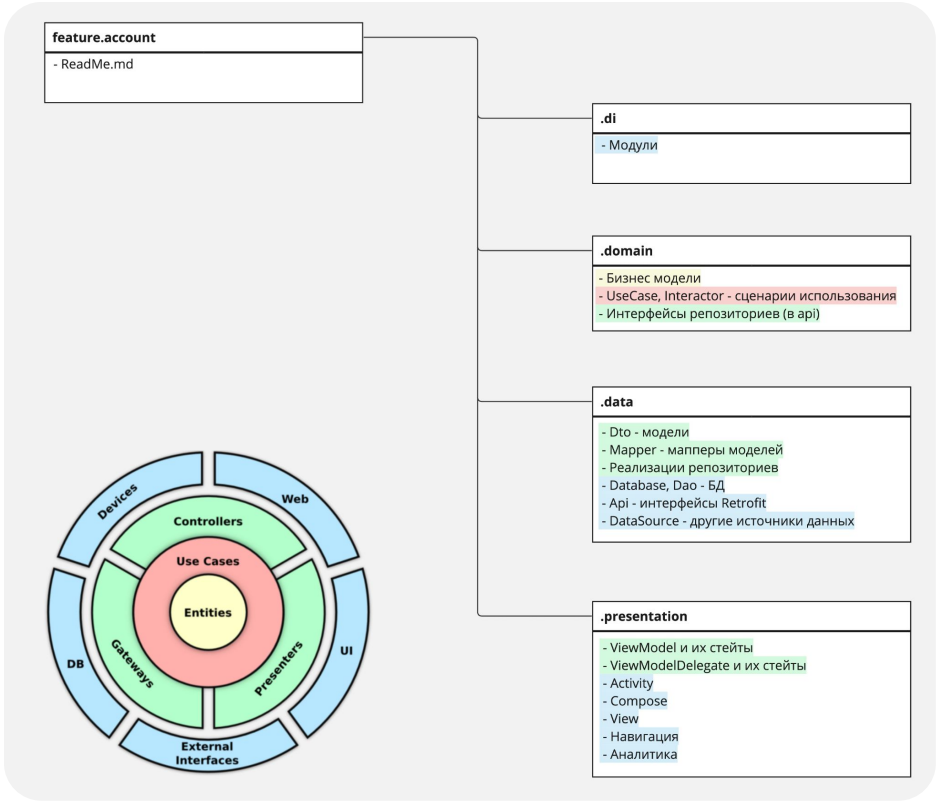
- Слабые зависимости
- Не ухудшает время сборки
- Легко выделять module owners



Архитектура

TTM | Scalability | Stability

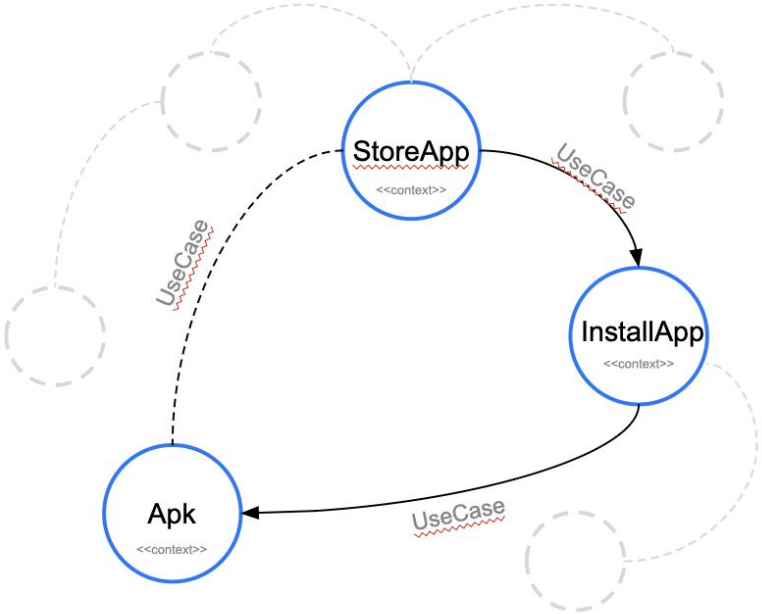
- Простая в применении
- Тестируемая
- Clean MVVM State подход

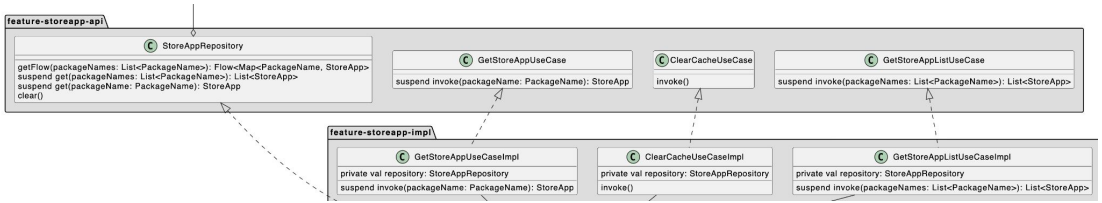


Бизнес-логика

TTM Scalability

- Отражает продукт
- Легко понимается
- Domain-driven design подход
- Переиспользуется





Время сборки проекта

TTM

- Максимум кэша у зависимостей
- Нет лишних конфигураций
- Применены лучшие практики по оптимизации Gradle

Прозрачность Контроль

План исправления техдолга

Группировка проблем

Собрать проблемы

Заводим проект-задачу аудита

Находим технические проблемы,
заполняем по шаблону в задаче

~2-4 недели

Группировка

Фактор
роста техдолга

Расставляем приоритеты

~1 неделя

Группировка проблем по факторам и приоритетам

SCALE P0..30

- Фактор Scale означает, что техдолг растёт
- Приоритеты **(P) - 0..30** проставляться в зависимости от наличия фактора и влияния на метрики
- Объем работ помогает понять насколько большая будет задача

Проблема	Влияние на метрики	Зона	Факторы	Приоритет	Предполагаемый объем работы	Задача	Описание
☑ Лишние конфигурации Gradle	BUILD TIME	TOOLS	SCALE	0	XS	~~RCORE-538~~ - [AND] Убрать лишние проекты из gradle ЗАКРЫТ	› Описание › Важность
☑ Eventbus, который используется повсеместно в проекте, включая широкое использование для навигации в UseCase	STABILITY TTM	ARCH	SCALE	10	XL	~~RUSTORE-630~~ - [AND] Избавиться от навигации в юзкейсах ЗАКРЫТ	› Описание › Важность
☑ Уязвимость Lce	STABILITY	ARCH		30	M	☑ ~~RCORE-1880~~ - Убрать IceFlow из репозиториев в общих случаях ЗАКРЫТ (старая задача)	› Описание › Важность
☐ Костыль с блокирующей навигацией	STABILITY TTM	ARCH		25	M	RUSTORE-5261 - [AND] Избавиться от костыля блокирующей навигации НОВЫЙ	› Описание › Важность
☑ Модули testFixtures	BUILD TIME	TOOLS	SCALE	0	S/M	☑ ~~RUSTORE-8655~~ - Отказ от использования testFixtures в проекте ЗАКРЫТ	› Описание › Важность
☑ Интеракторы вместо юзкейсов	TTM	ARCH	SCALE	15	XL	Зафиксированы требования к БЛ в рамках Логика на domain слое Выделять в отдельный эпик переработку имеющихся юзкейсов нецелесообразно	› Описание › Важность
☑ Логика в конструкторе	STABILITY TTM	ARCH	SCALE	5	L	☑ ~~RUSTORE-397~~ - Вынос инициализации VM из init {} ЗАКРЫТ	› Описание › Важность

Составляем план и действуем

Собрать проблемы

Заводим проект-задачу аудита

Находим технические проблемы,
заполняем по шаблону в задаче

~2-4 недели

Группировка

Фактор
роста техдолга

Расставляем приоритеты

~1 неделя

Roadmap

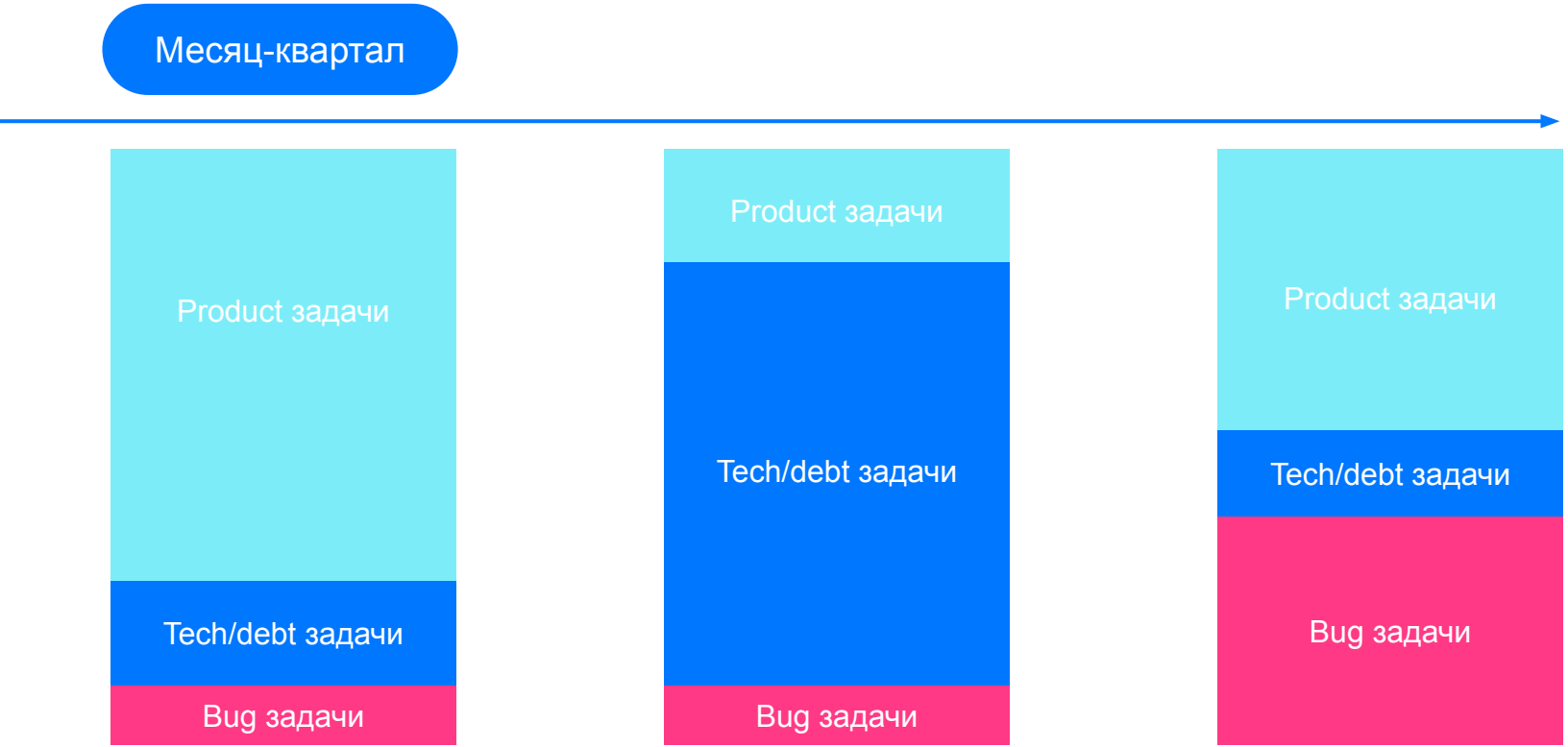
OKR

Gant

~1 неделя

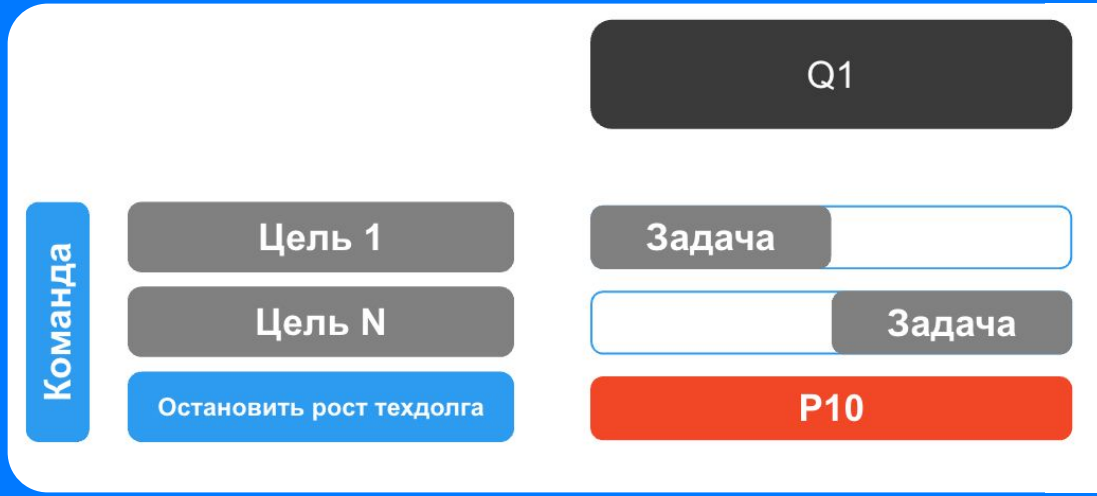
Составляем план и действуем

Баланс задач техдолга в бэклоге



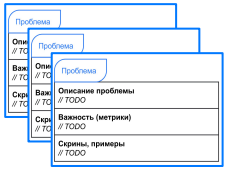
Преимущество данного подхода

- Не надо останавливать разработку фич или других задач
- Гарантия выполнения и прозрачный контроль
- Выполняете самое приоритетное



Убираем последовательно весь собранный техдолг

Собрать проблемы



~2-4 недели

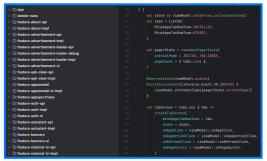
Группировка

~1 неделя

Roadmap

~1 неделя

Фикс p10



Q1

Фикс p20-30

p10

Закрыт

Qn

Мы убрали
весь техдолг?

Итоги борьбы с техдолгом в RuStore

01 За 3 квартала полностью остановили рост техдолга в проекте

02 Меньше чем за 1 год, мы сделали масштабируемую архитектуру с 300 модулями на проекте

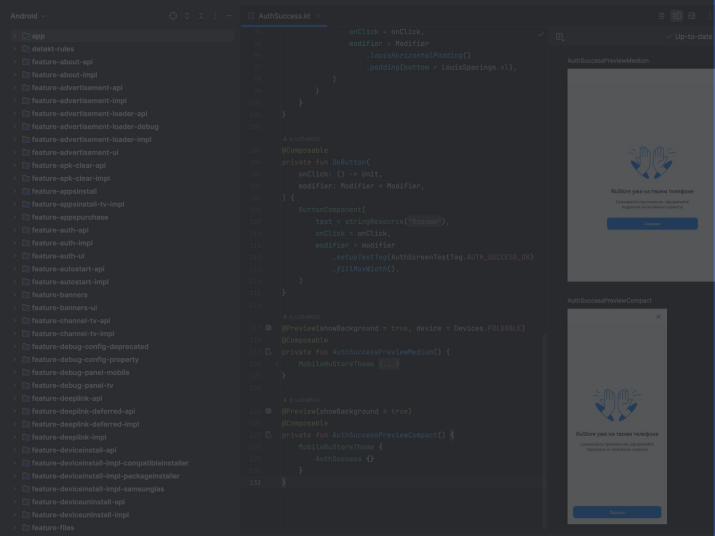
03 Время сборки на холодную занимает ~5 минут

04 Рефакторинг кода не отнимает время разработки фичей

90% проблем

Польза

- 20 команд
- 30 человек на проекте
- Современная архитектура
- Топовый стэк
- Compose
- Нет зависимостей от Google Service чел. на проекте
- Flow
- DDD
- Scalable module structure



Содержание

Как победить техдолг у вас?

01 Помните! Техдолг может принести урон вашему продукту!

02 Сначала определитесь с метриками, далее делайте анализ

03 Выбирайте стратегию изменений решающую ваши задачи

04 Проставьте приоритеты, потом планируйте

05 Для фикса техдолга, используйте баланс бэклога, а не квоты (они не работают)

RuStore

Контакты

Группа в Tg для
разработчиков

Страница RuStore
для разработчиков
dev.rustore.ru