FSD при переходе на новый стек

Михаил Дмитриевский

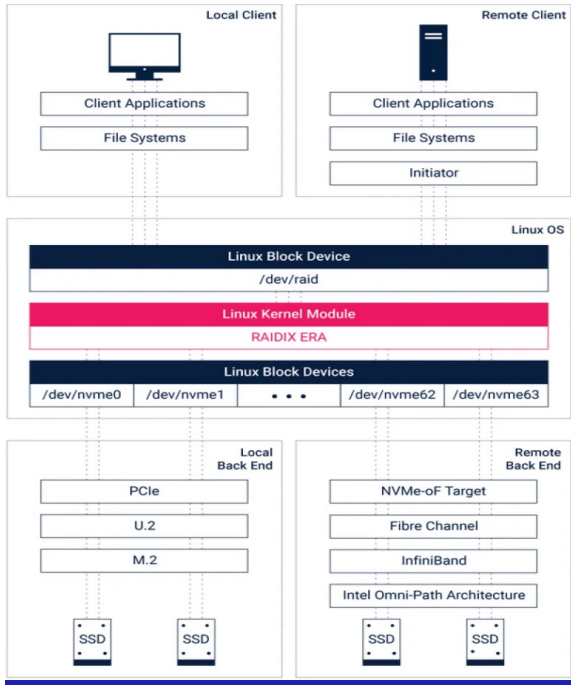
YADRO, департамент СХД, проект Tatlin.FLEX

- Фронтенд-разработчик
- Agile-lead команды UI
- Lead фронтенд-сообщества YADRO

О продукте

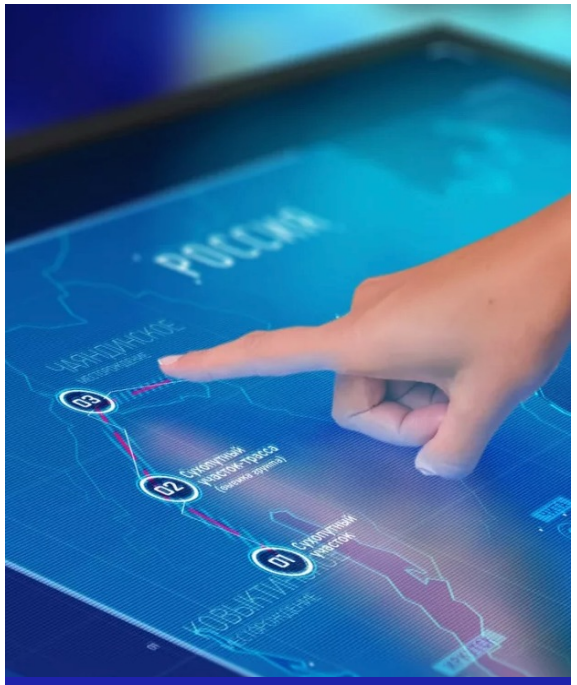
Tatlin.FLEX (.ONE)

Многофункциональная СХД для среднего и малого бизнеса



Raidix

ПО для построения высокопроизводительных систем хранения данных



UI

Веб-сервис для управления СХД

Задача – смена стека

Ускорение сервиса

Упрощение развития и поддержки

Облегчение onboarding сотрудников

Глобальные проблемы

I. Тяжелая поддержка и развитие

II. Спагетти-код со спутанной БЛ

III. Отсутствие разработчиков на текущий стек

О сервисе

Текущий стек

- Backbone
- Marionette
- Handlebars

Кодовая база

- Монолит
- View, Models (& Collection)
- Шаблоны, хелперы, утилиты

Особенности

- 122 сущности
- 3 роли
- 29 типов лицензий
- Очередь команд

Реализация модели и коллекции

```
/* Model */

import Backbone from 'backbone’;

export const Raid = Backbone.Model.extend({

        urlRoot() { … },

        defaults() { … },

        ….
};
```

// 919 строк

```
/* Collection */

import Backbone from 'backbone';

export const Raids = Backbone.Collection.extend({

        model: Raid,
        comparator: 'name’,

        initialize { … },

        ….
};
```

// 300 строк

Реализация таблицы и её применение

```
/* Реализация */

import Marionette from 'marionette’;
…

export const TableView = Marionette.ItemView.extend({
   template: tableTemplate ,

   initialize(...rest) { … },

   onRender() { … },
   ….
};
```

// 2100 строк

```
/* Применение */

import Backbone from 'backbone’;
…

export const RaidTable = TableView.extend({
   template: raidsTableTemplate,
   collection: raids,

   initialize(...rest) { … },
   ….
};
```

// 2300 строк

Первая попытка

Промежуточный слой

Backbone + MST

1

2

Перенос моделей

Реализовали 80 %

Внедрение

Попытка соединения Backbone и нового фреймворка

3

4

Ретроспектива

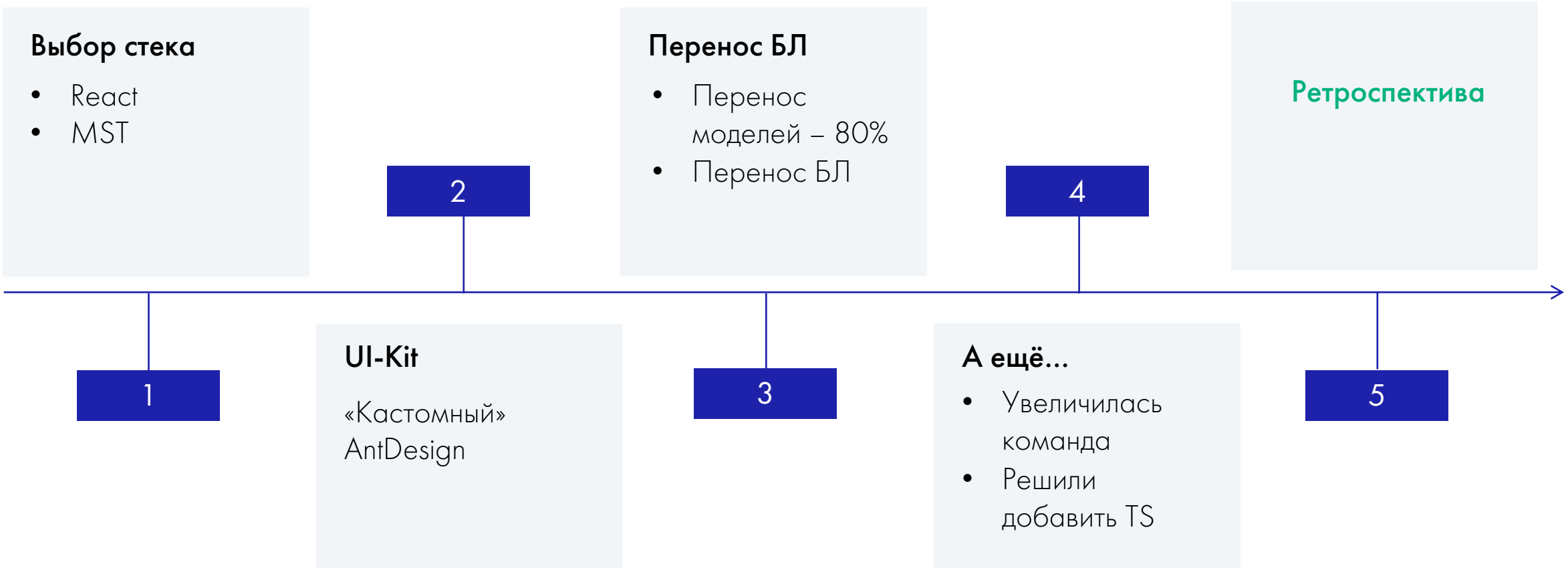
Глобальные проблемы. Что изменилось?

I. Тяжелая поддержка и развитие

II. Спагетти-код со спутанной БЛ

III. Отсутствие разработчиков на текущий стек

Вторая попытка



Глобальные проблемы. Что теперь изменилось?

I. Тяжелая поддержка и развитие

II. Спутанная БЛ

III. Отсутствие разработчиков на текущий стек

Реализация таблицы на React'е и её применение

```
/* Реализация */

// … imports

export  const CustomTable: FC<…> = ( props ) => {
        …
        return (

                <StyledTable>
                        <Table … />
                </StyledTable>

        );
};
```

// 105 строк + some modules

```
/* Применение */

// … imports

export const RaidsTable: FC<…> = ({ models, button
}) => {

        const { raids } = useStore();
        const columns = columnsRaid();
        const collection = models || raids.getModels();

        return <CustomTable …/>
};
```

// 24 строки + config 200 стр.

Глобальные проблемы. Что теперь изменилось?

I. Тяжелая поддержка и развитие

II. Спутанная БЛ

III. Отсутствие разработчиков на текущий стек

Варианты новой архитектуры

Atomic

- Хорошо для дизайна
- Нашу запутанность БЛ не решает

Micro Frontend

- Не тяжёлый сервис, сложность – БЛ СХД
- Over-engineering

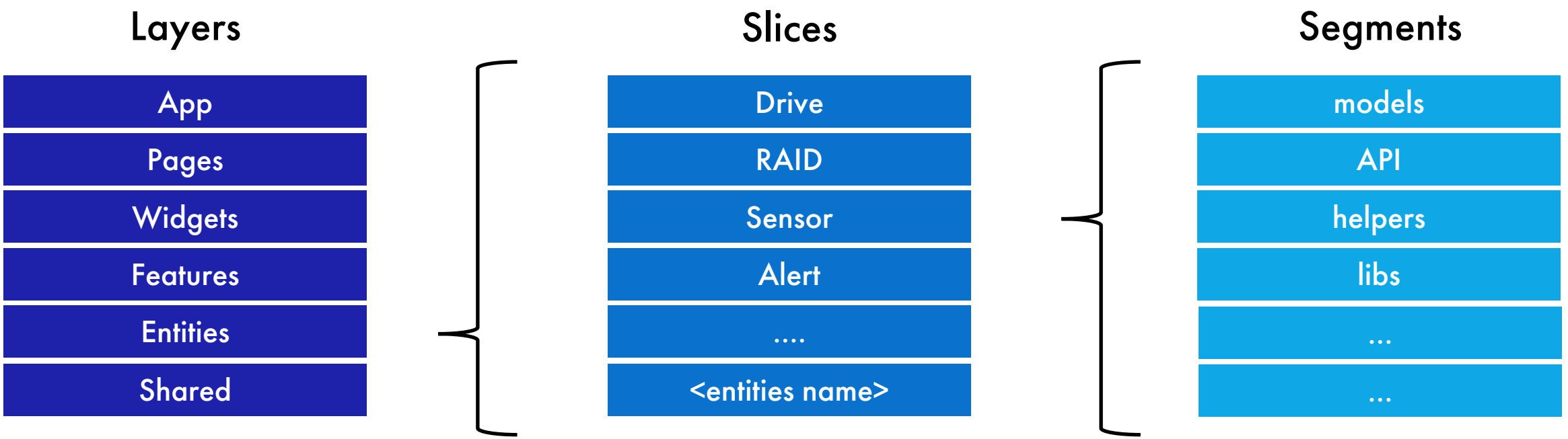
Modules

- Модули могут переиспользоваться в другом модуле
- Что делать с бизнес-сущностями?

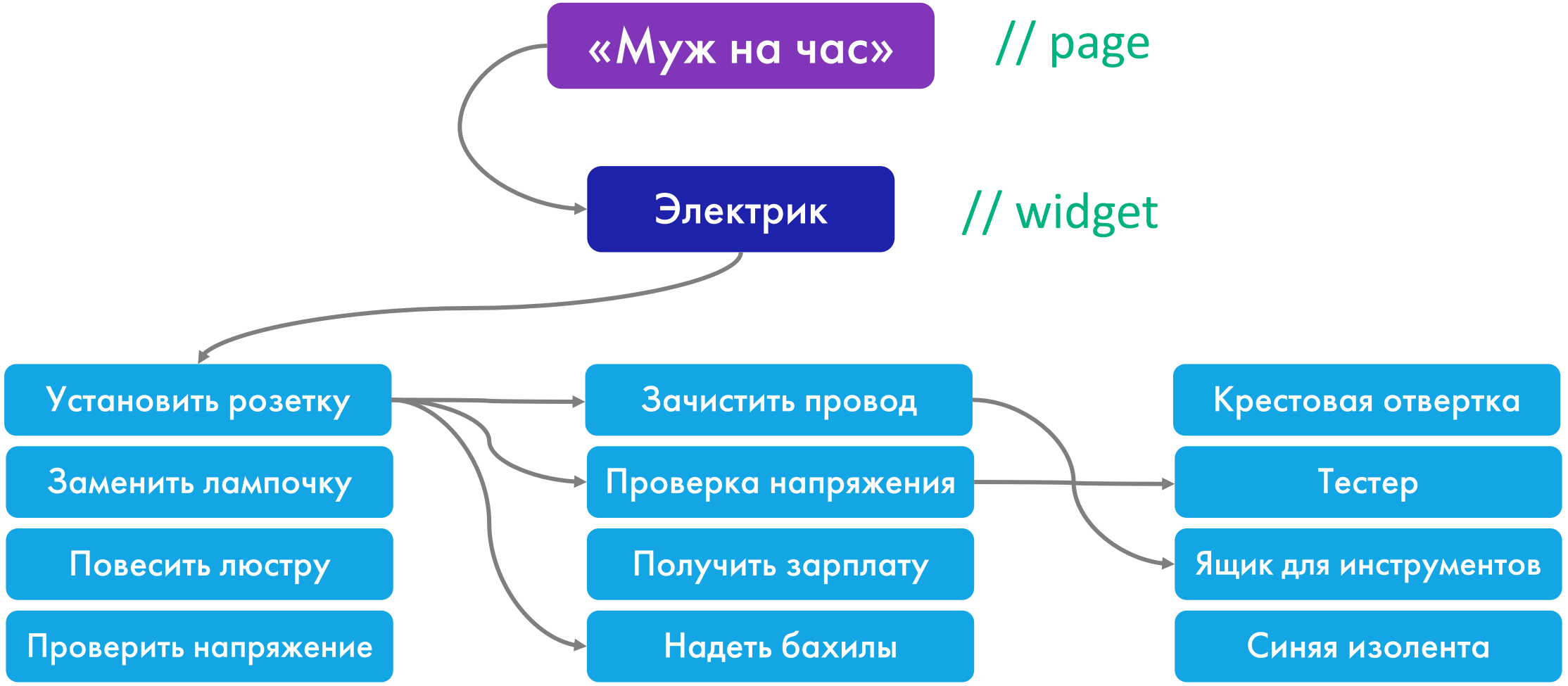
FSD

- Количество бизнес-сущностей – 122
- Взаимосвязь моделей, ролей, лицензий, очереди команд

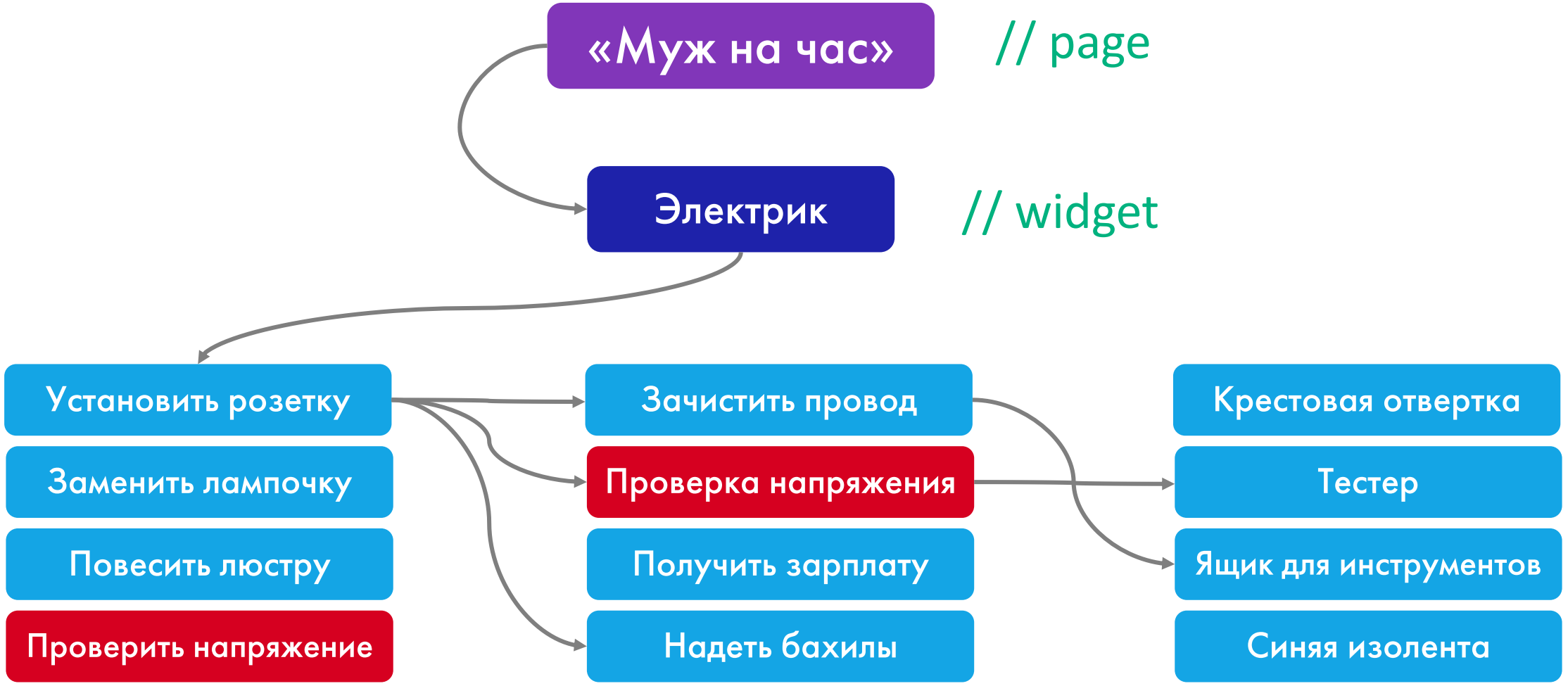
Коротко про FSD



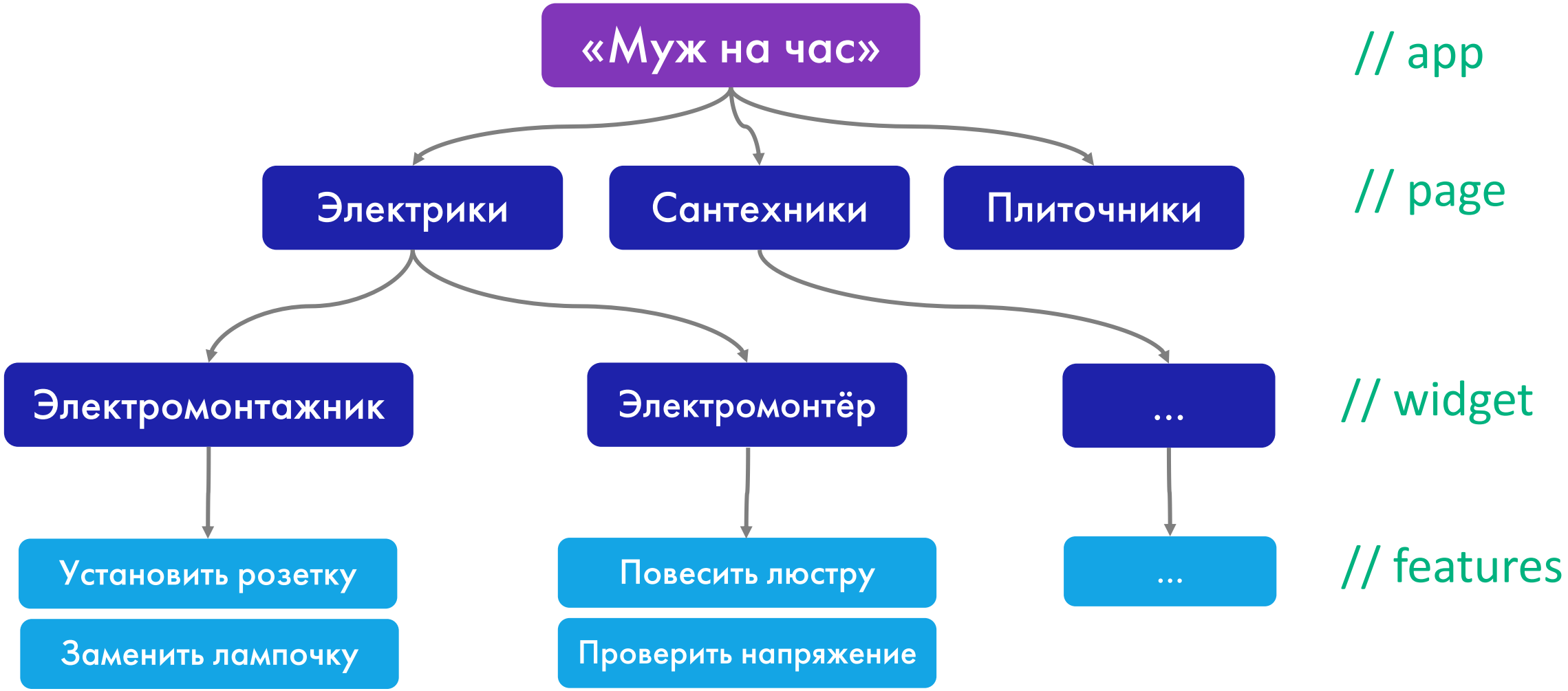
«FSD на пальцах»



«FSD на пальцах»



«FSD на пальцах»

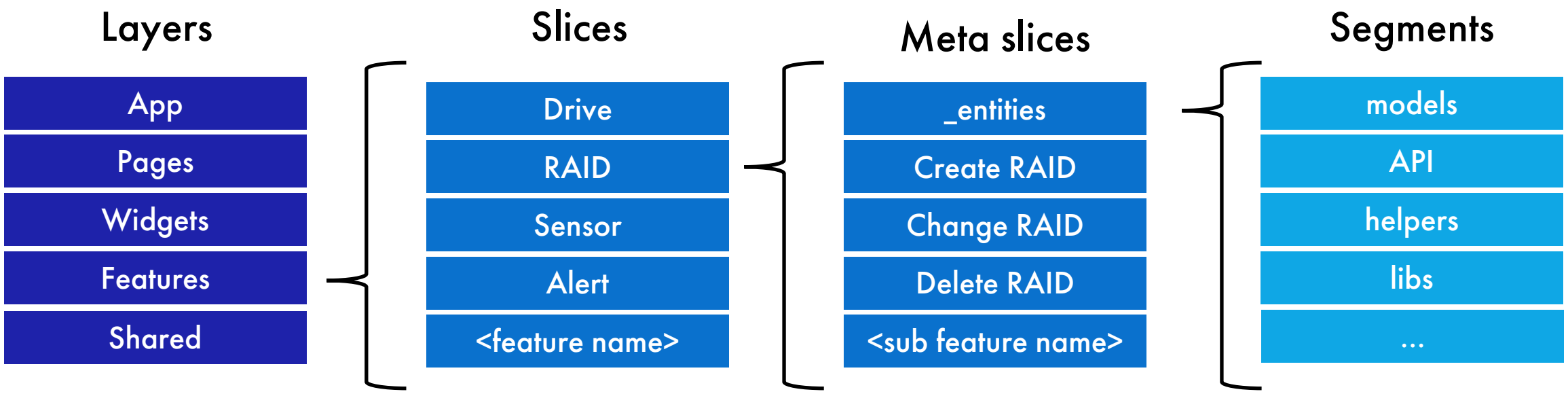


Важное дополнение

Направление взаимодействия

1 фича – 1 интерфейс

Всё хорошо, но как у нас



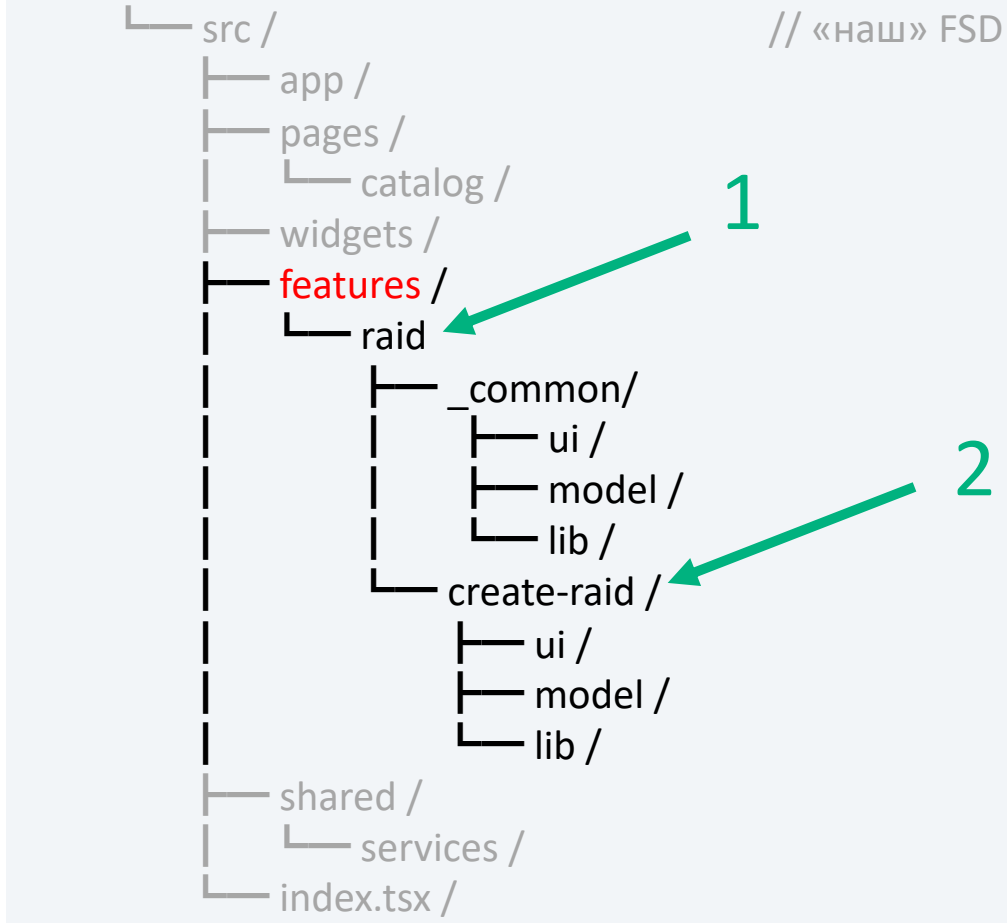
Зачем нам sub_features?

Объединение общих компонентов или логики, характерных только для конкретной фичи

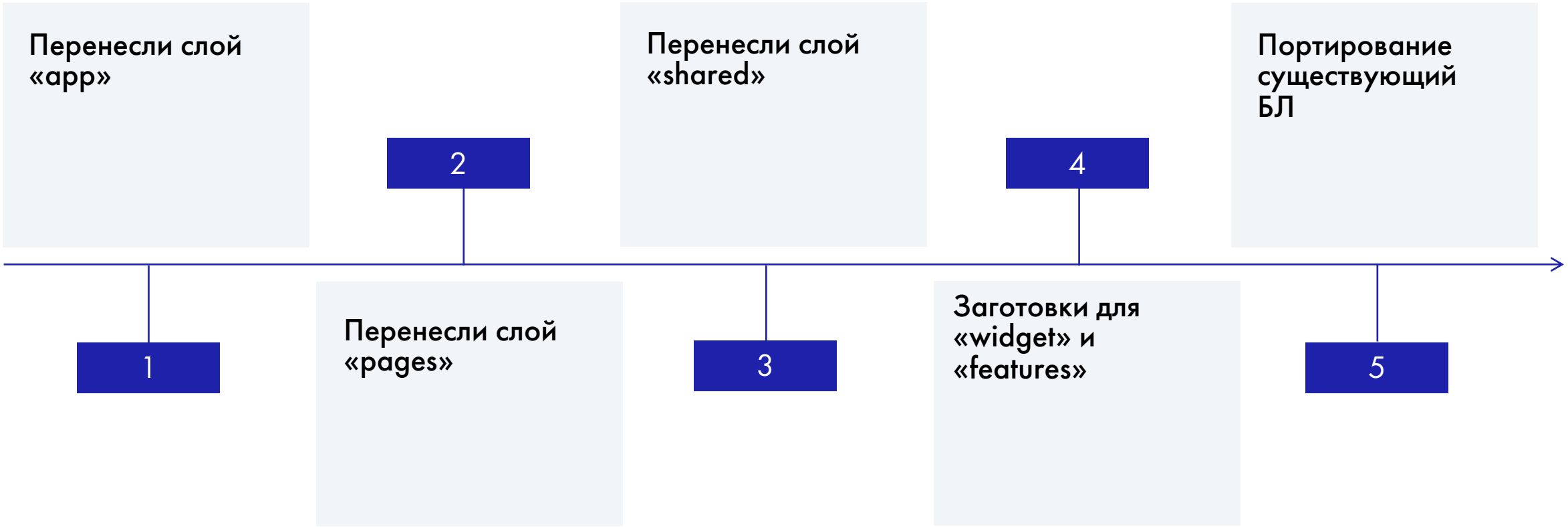
Не засоряется архитектура, проще визуальная навигация

Сравнение реализаций в директориях

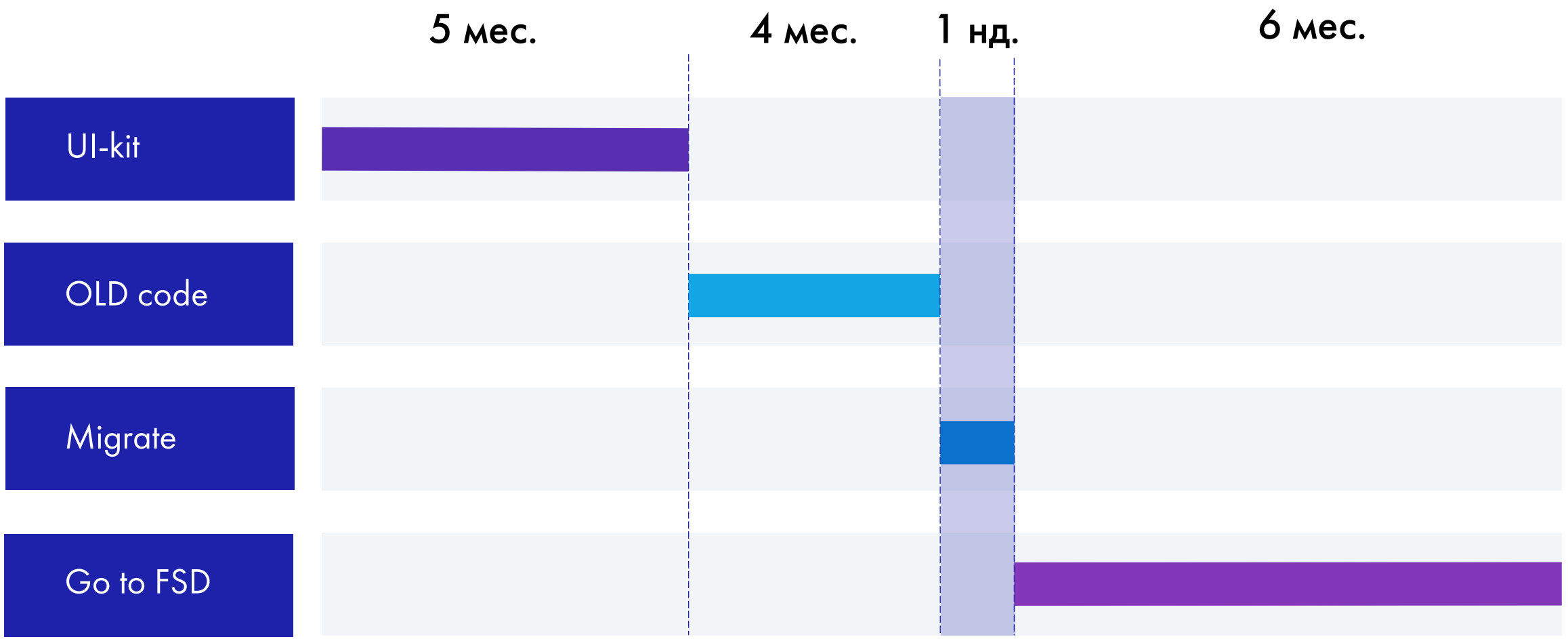
```
└── src /                                  // FSD
    ├── app /
    ├── pages /
    │   └── catalog /
    ├── widgets /
    ├── features /
    │   └── add-to-card
    │       ├── ui /
    │       ├── model /
    │       └── lib /
    ├── entities /
    │   └── cart /
    ├── shared /
    │   ├── components /
    │   ├── lib /
    │   └── services /
    └── index.tsx /
```



Процесс переход



Трудозатраты

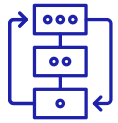


Договорённости в команде

Разработчик отвечает за свою features

Перенос слоя – последовательная задача для одного разработчика

Выделился слои widget и features – переход всей команды на FSD



Логический конец по текущей задаче – портирование БЛ на FSD

Минусы, подводные камни и "боли"

Тяжело понять архитектуру

Тяжело понять, **что́** является "фичёй"

Горизонтальный импорт

Не все фичи можно объединить в виджетах

Тяжело "разорвать" бизнес-логику

Человеческая лень

Плюсы и выводы

Разработка

- Уменьшили связанность БЛ
- Независимость фичей
- Меньше гит-конфликтов и выше скорость review
- Разработчик в контексте фичи
- Итерационное внедрение

Управление

- Плавность перехода - 4 месяца за 1 неделю
- Задачи максимально минимальные
- Не важна последовательность задач
- Проще определять объем
- Проще масштабировать коллектив
- Легче onboarding и старт работы

Глобальные проблемы. Ну а сейчас?

I. Тяжелая поддержка и развитие

II. Спутанная БЛ

III. Отсутствие разработчиков на текущий стек

Небольшие советы и «лайф-хаки»

Когда использовать?

- Проект с «компонентным фреймворком»
- «Админки» (для СХД)

Будущее и настоящее

- Не думать на «сто» шагов вперёд
- Регулярный микрорефакторинг

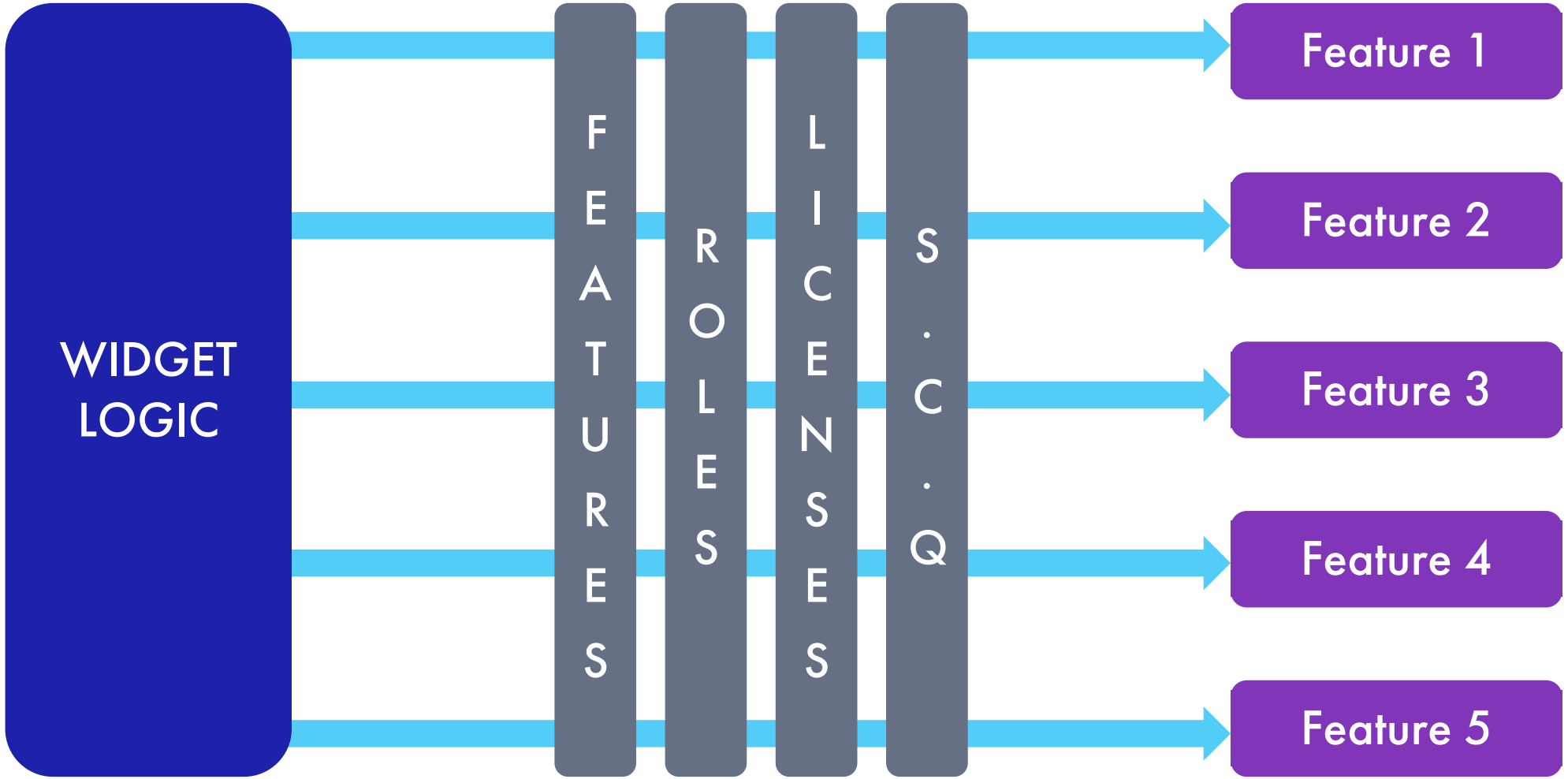
Features

- Логически законченная бизнес-функциональность
- SMART

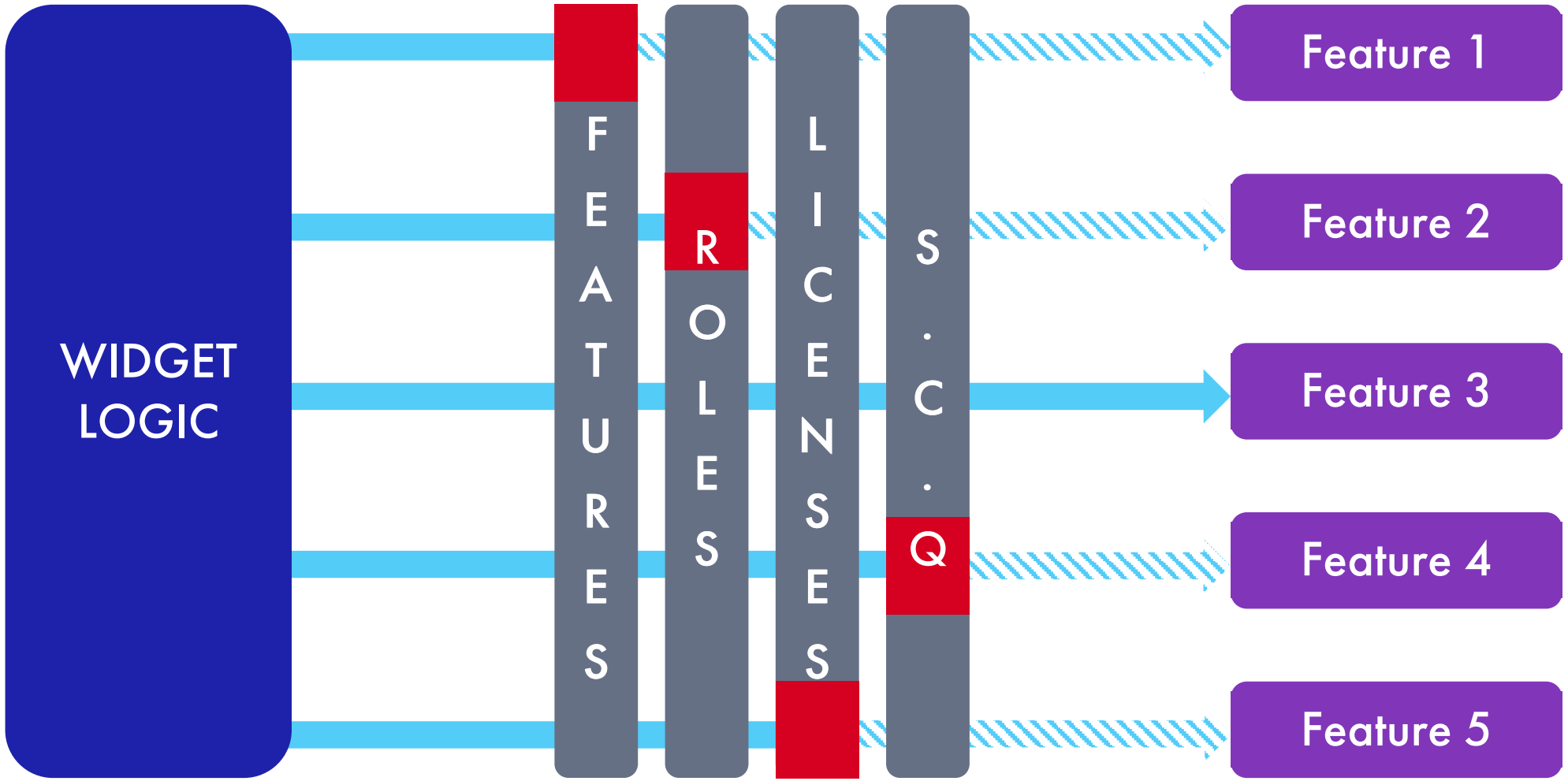
Widget

- Занимается управлением «фичёй»
- Понимает, когда есть «фича», а когда её нет

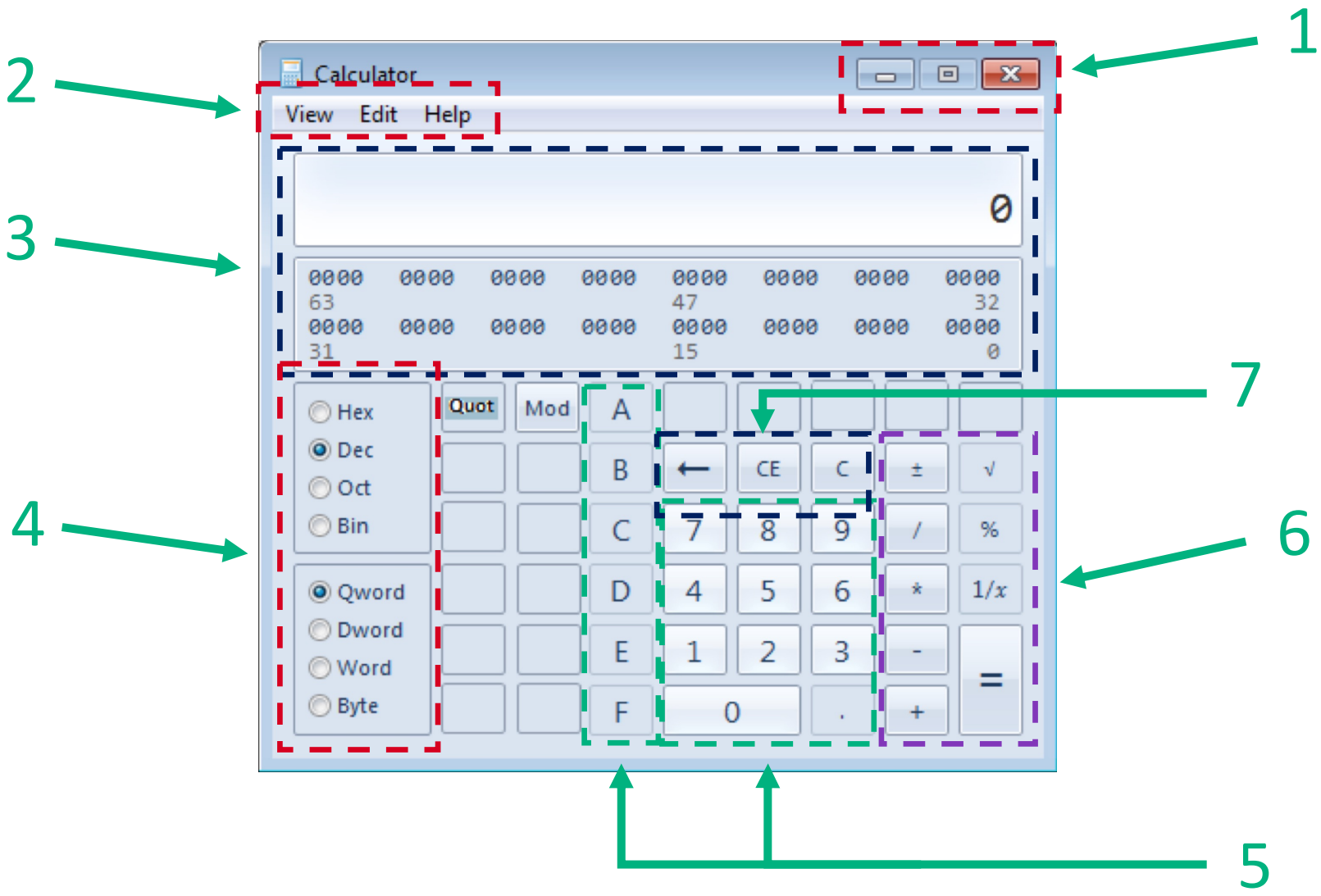
Бонус 1 – визуализация взаимодействия



Бонус 1 – визуализация взаимодействия



Бонус 3 – «FSD на примере»



Полезные ссылки

Документация FSD

eslint-plugin-fsd

sonar

Ссылка на презентацию

Всем спасибо! =)


yadro.com