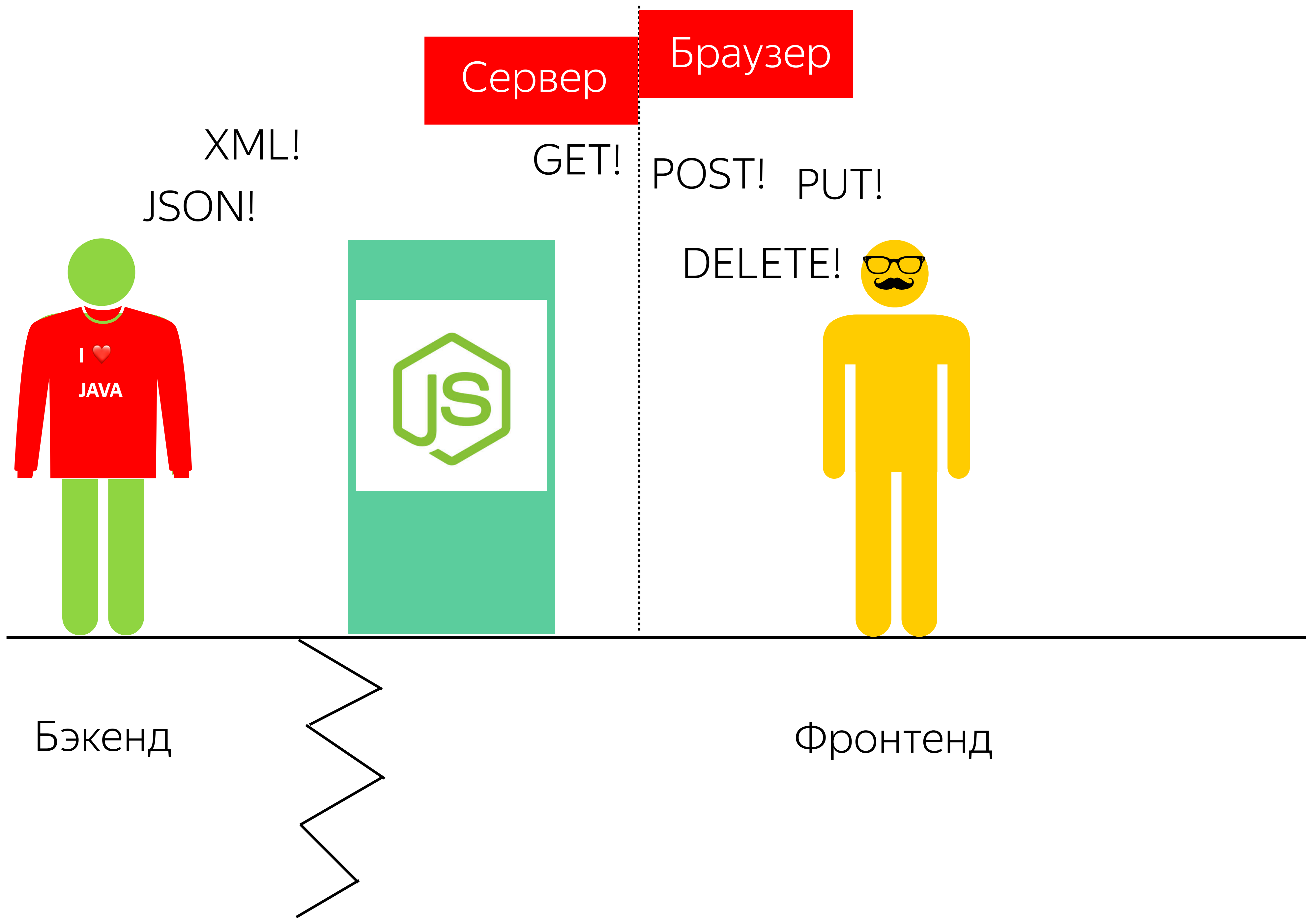


Яндекс Деньги

Рефакторинг платежного процесса Яндекс.Денег

Илья Кашлаков



Как понять что
рефакторинг нужен?





В коде разбирается
один разработчик





Нет четкой структуры
кода





Добавление каждой
новой фишки дорожке



Подготовка к рефакторингу




В новом решении должна быть возможность

- › Легко добавлять новые фичи
- › Переиспользовать часть логики для разных платежей
- › Простая для понимания реализация
- › Длительная поддержка



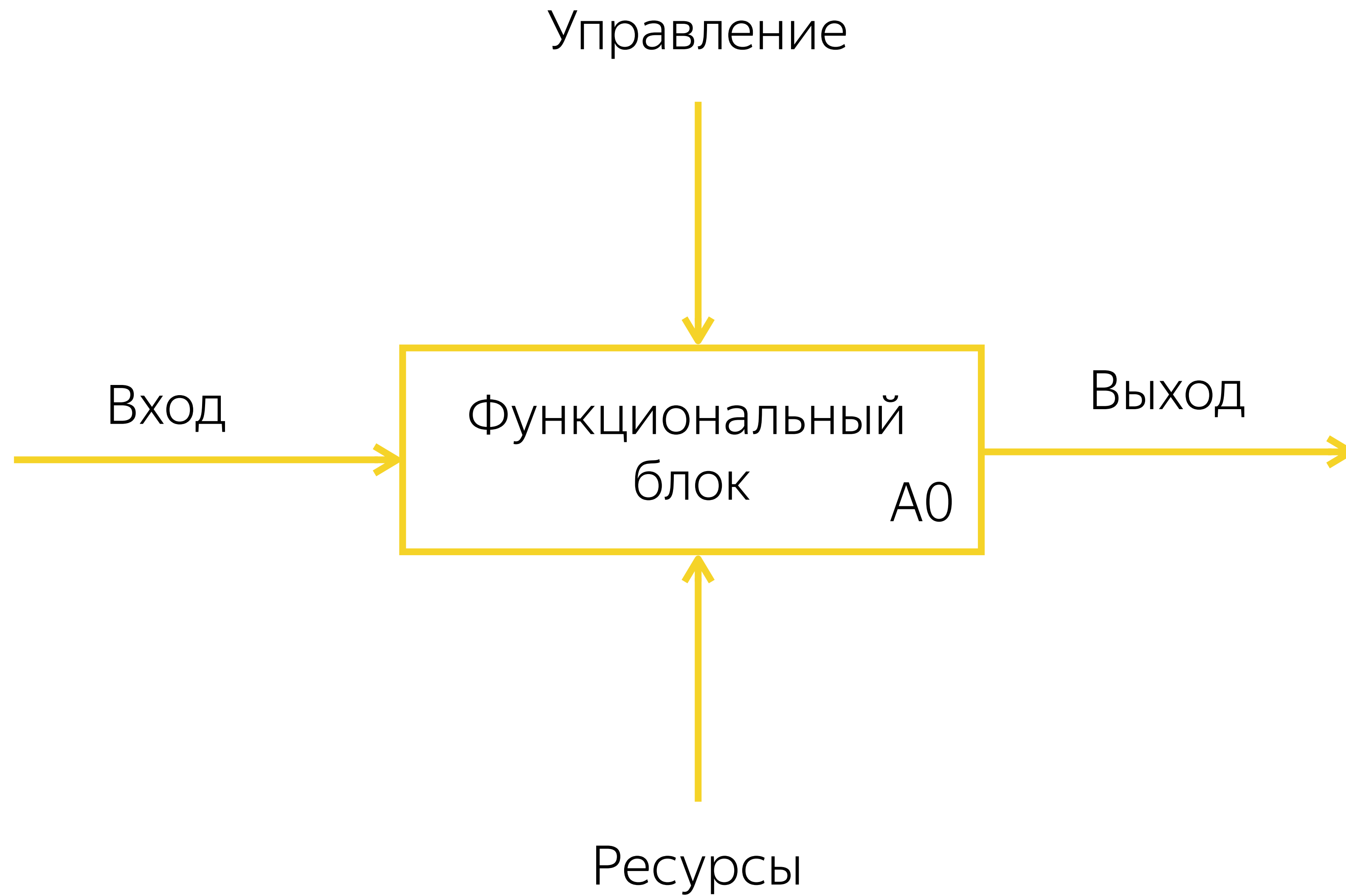
Кто участвовал разработке

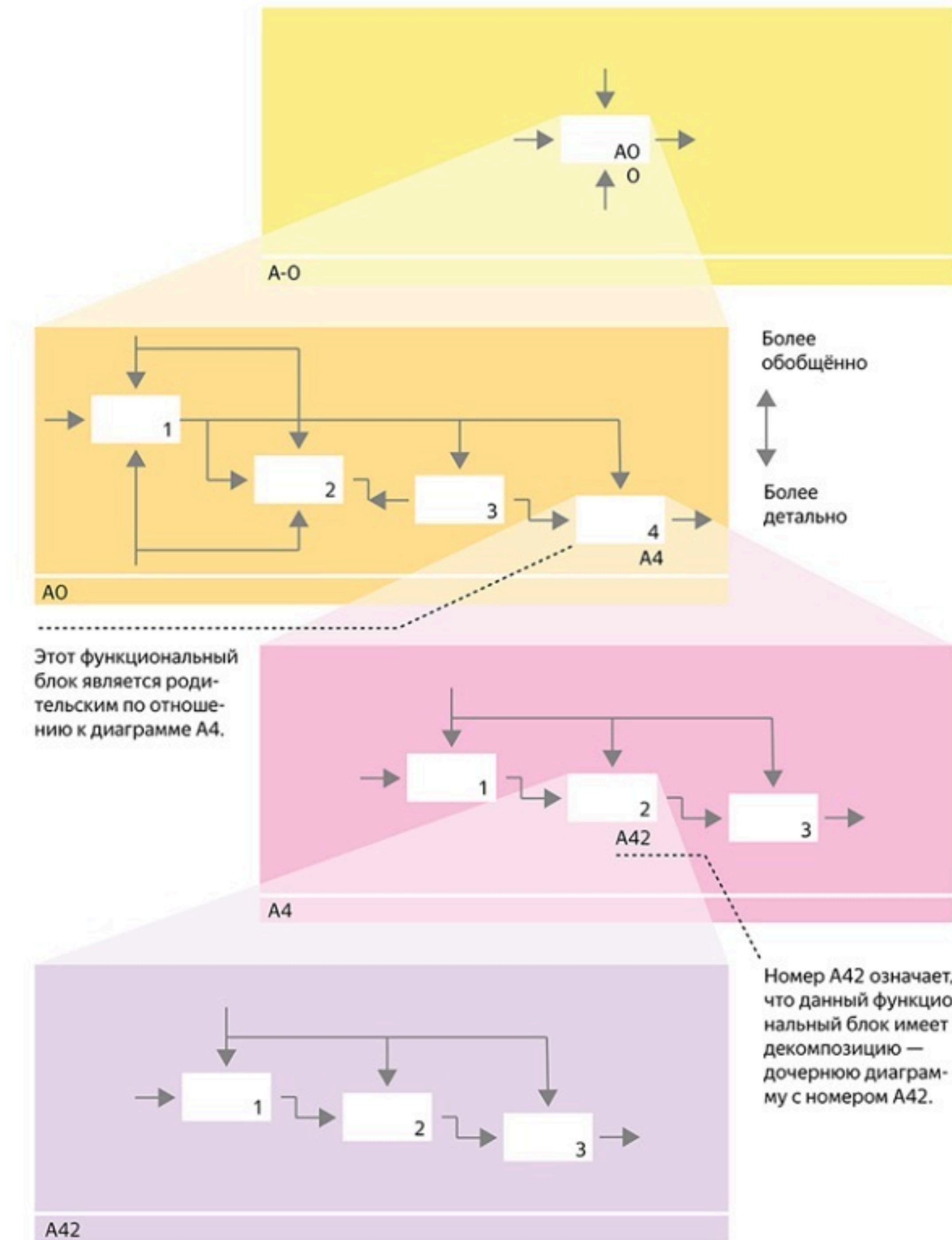
- › Команда для разработки нового решения
- › Рабочая группа для оценки принятых решений



Теоретическая основа
очень важна

IDEF0





В новом решении мы используем идеи IDEF0



Реализация





ProcessFlow

```
module.exports = new ProcessFlow({
  // Указываем, какой функциональный блок отвечает за вход в процесс
  initialStage: 'checkUserName',
  // Описываем выходы из процесса
  finalStages: [
    'userCheckedSuccessful',
    'userCheckFailed'
  ],
  stages: {
    checkUserName($flow, inputData) {
      // логика проверки имени пользователя
      $flow.transition('checkUserBalance', inputData);
    },
    checkUserBalance($flow, inputData) {
      // логика проверки баланса пользователя
      $flow.transition('checkUserPhone', inputData);
    },
  },
});
```



```
checkUserPhone($flow, inputData) {  
    const phone = inputData.operatorCode + inputData.number;  
    checkPhoneProcess.start(phone, {  
        // описываем поведение в точках выхода процесса проверки телефона  
        phoneValid() {  
            $flow.transition('userCheckedSuccessful');  
        },  
        phoneInvalid() {  
            $flow.transition('userCheckFailed');  
        }  
    });  
}  
}  
})
```

```
/**
 * Функциональный блок для проверки имени пользователя
 * @param {Object} $flow служебный объект, экземпляр текущего процесса, позволяющий
управлять переходами от блока к блоку
 * @param {Object} inputData входящие данные
 * @param {Object} inputData.userName имя пользователя
 */
const checkUserName = ($flow, inputData) => {
  if (inputData.userName) {
    // Переходим к следующему функциональному блоку
    const outputData = {
      userName: inputData.userName,
      isUserNameValid: true
    };
    $flow.transition('doSomethingElse', outputData);
    return;
  }
  $flow.transition('checkFailed', inputData);
}
```




ПОНЯТНЫЙ КОД - ЭТО
ограничения

Ограничения

- › Не более 6 функциональных блоков на каждом уровне
- › Нижний предел - 3 функциональных блока
- › Количество выходов из одного блока ограничено
- › Один функциональный блок - одно действие

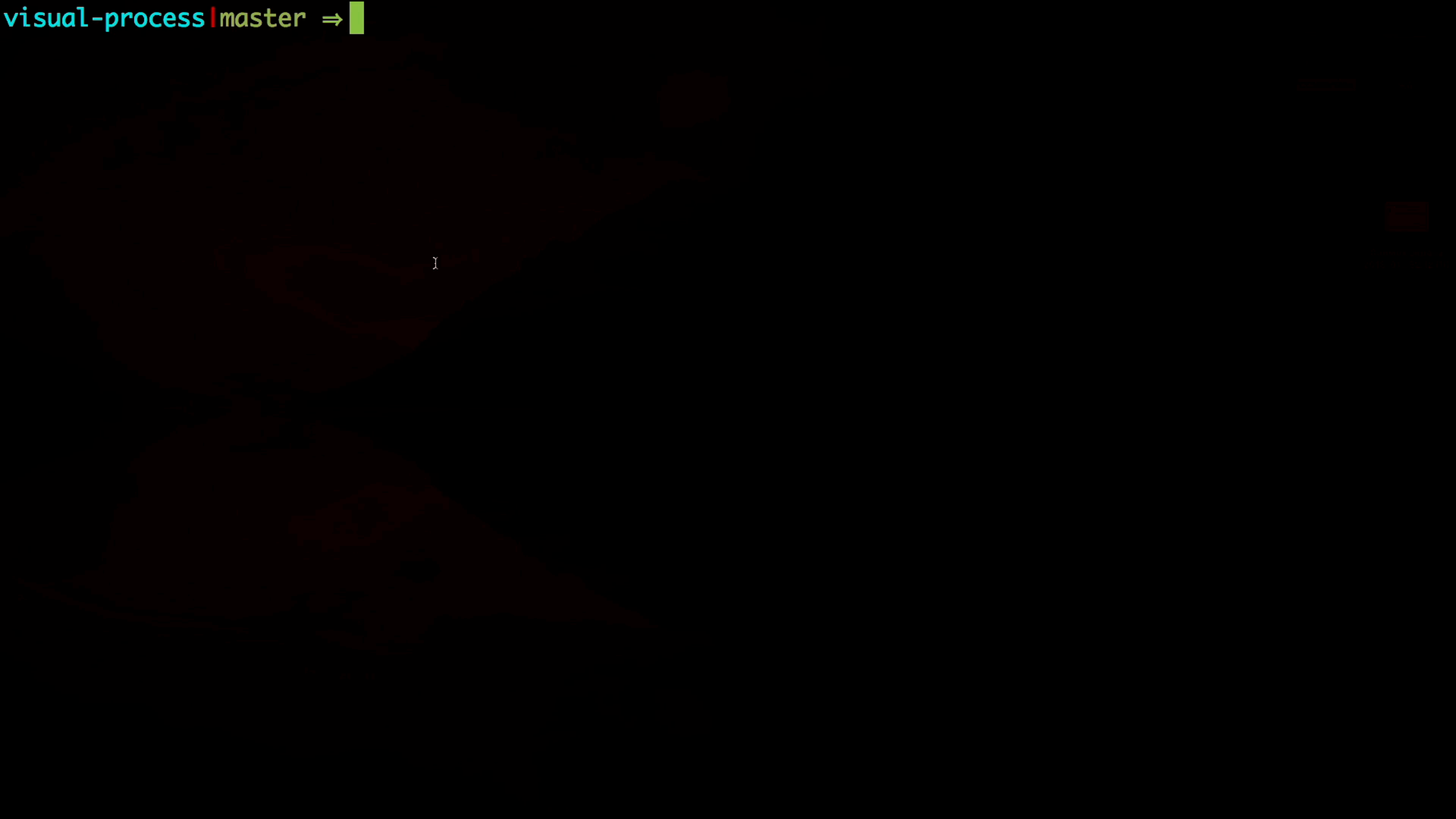
Визуализация





Блок схема процесса
строится по коду

visual-process | master =>

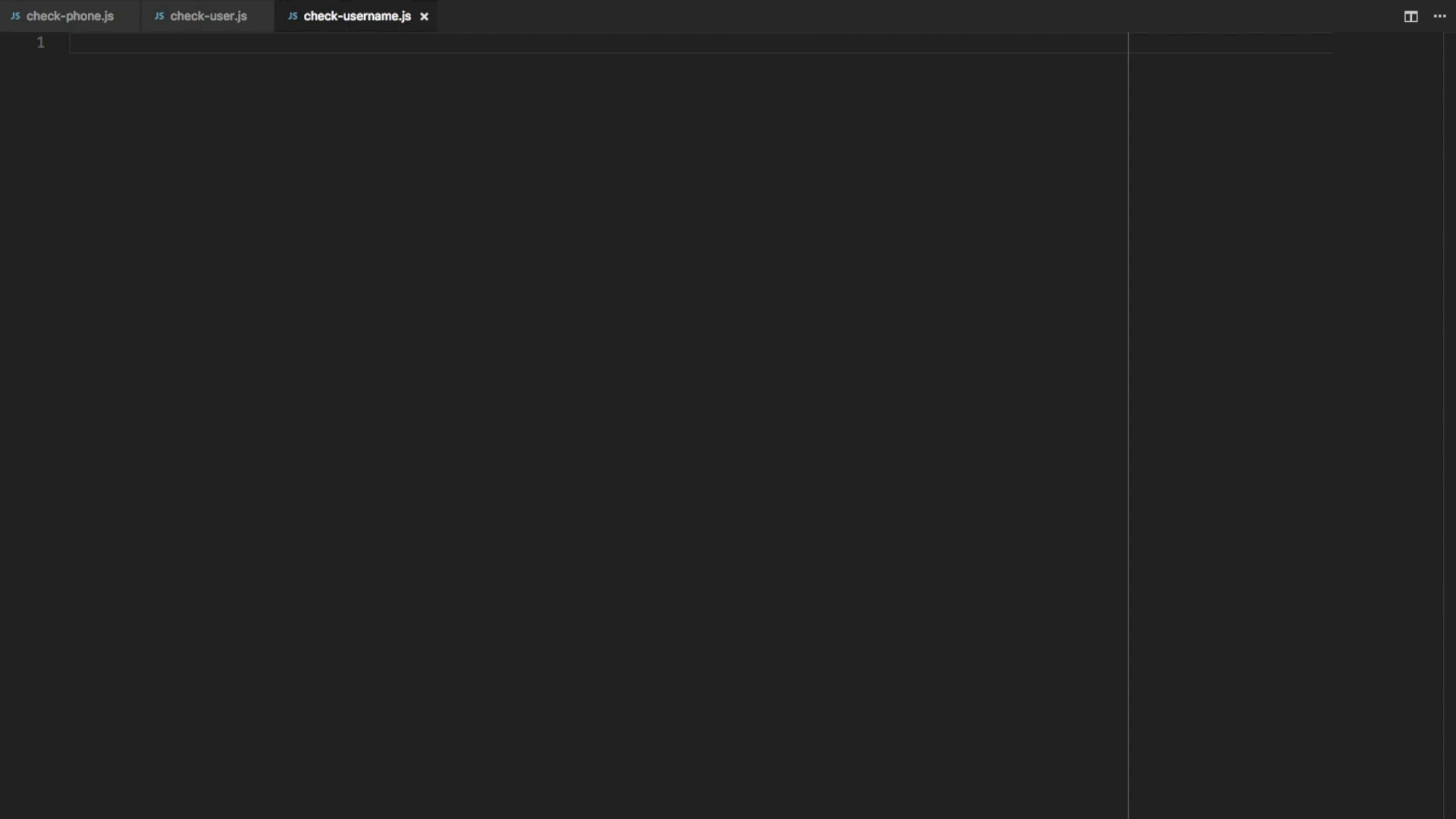


Что в итоге?



Итоги

- › Важны не технологии, а подходы
- › Код должен быть как можно проще
- › Единая теоретическая основа - залог хорошей поддержки



Почитать ещё

<http://bit.ly/2FO7mRu>



Спасибо за внимание

Илья Кашлаков

Ведущий разработчик

Яндекс.Деньги



@mrkashlakov



@mr_kashlakov