Быстрое прототипирование функциональных макетов UI на Lua и Mermaid.js

Александр Гладыш
@agladysh

Lua in Moscow / DevConf 2017

План

1. Кейс
2. Подходы к проектированию
3. Enter the Mermaid
4. Шаблоны Lugram
5. В завершение
6. Вопросы?

Обо мне

- Программист
- Сейчас основном занимаюсь управлением
- Пишу на Lua с 2005-го года

Кейс

- Огромная профессиональная энтерпрайз-система
- обновляется от 20-летнего приложения для Windows
- до современного одностраничного веб-приложения.

Система огромна

Ни один человек не в состоянии при проектировании принять обоснованное решение по сложному вопросу в одиночку

- У экспертов в технологии нет видения нового продукта в целом
- ПМ и ПО не рисуют профессионально и у них не всегда есть действительно глубокое понимания аспектов технологии
- Дизайнер имеет лишь поверхностное понимание технологии

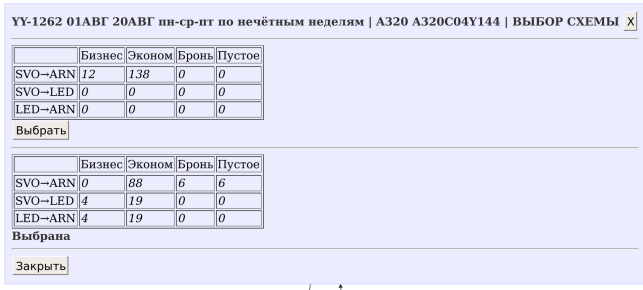
Процесс проектирования и дизайна UI

Для каждого ”экрана” в приложении:

- Концепция
- Функциональные макеты и (иногда) интерактивные эскизы
- Дизайн-макеты
- Вёрстка
- Реализация бизнес-логики

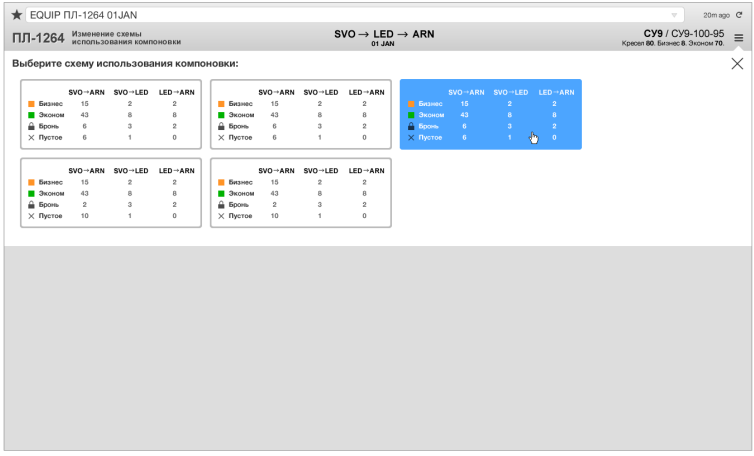
Функциональный макет

Что должно находиться на экране, как это будет РАБОТАТЬ?



Дизайн-макет

Как это должно ВЫГЛЯДЕТЬ?



Задачи

- Нужна диаграмма переходов между экранами приложения.
- Нужны функциональные макеты самих экранов.
- В принципе не важно как я получу эту диаграмму и макеты, до тех пор пока их легко сделать и изменить и есть хоть какие-то механизмы для повторного использования наработок.

Инструменты

- Photoshop (Krita, Gimp...)
- InkScape
- Google Documents
- Visio
- Balsamiq
- Sketch
- ...

Я — программист. Мне легче работать со структурированным текстом чем с изображениями.

Быстрее всего я работаю с клавиатурой, не трогая мышь.

Enter the Mermaid

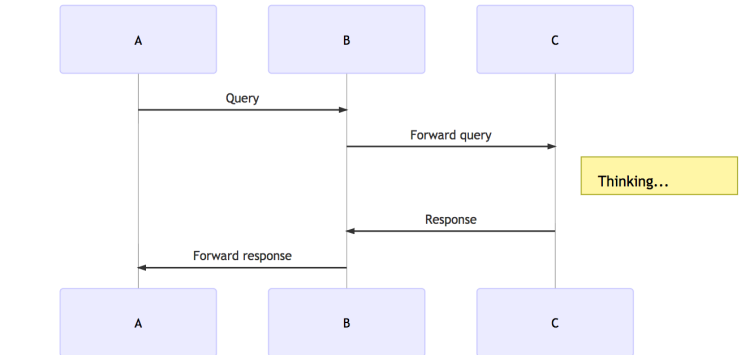
http://bit.ly/mermaid-editor

Enter mermaid diagram syntax here

```
sequenceDiagram
A->> B: Query
B->> C: Forward query
Note right of C: Thinking...
C->> B: Response
B->> A: Forward response
```

This webapp uses the mermaid library to generate graphs and sequence diagrams.
Syntax for sequence diagrams can be found here

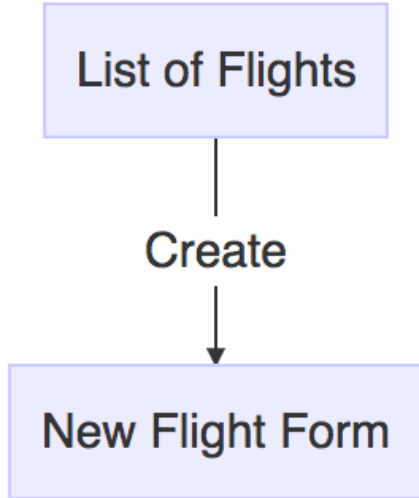
Generated diagram



LINK TO VIEW LINK TO EDIT DOWNLOAD SVG

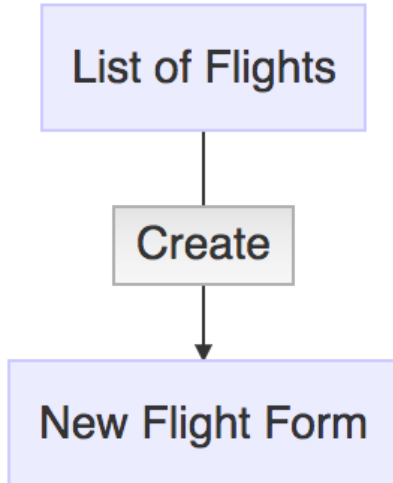
Диаграмма переходов между экранами

```
graph TD

list[List of Flights]
new[New Flight Form]

list-->|Create|new
```



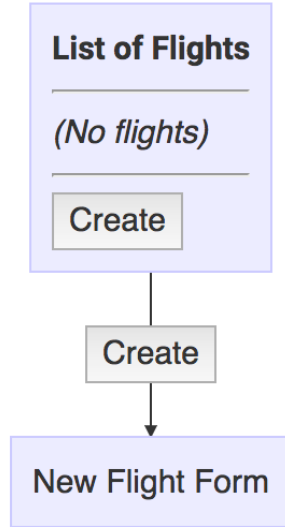
Диаграма переходов между екранами (II)

```
graph TD

list[List of Flights]
new[New Flight Form]

list-->|"
<button>Create</button>
"|new
```



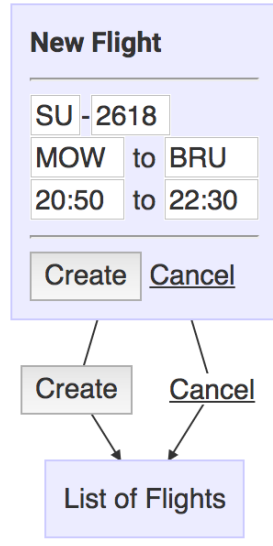
Функциональные макеты экранов

```
graph TD

list["<b>List of Flights</b>
    <hr>
    <i>(No flights)</i><hr>
    <button>Create</button>"]

new["New Flight Form"]
list-->|"
<button>Create</button>
"|new
```

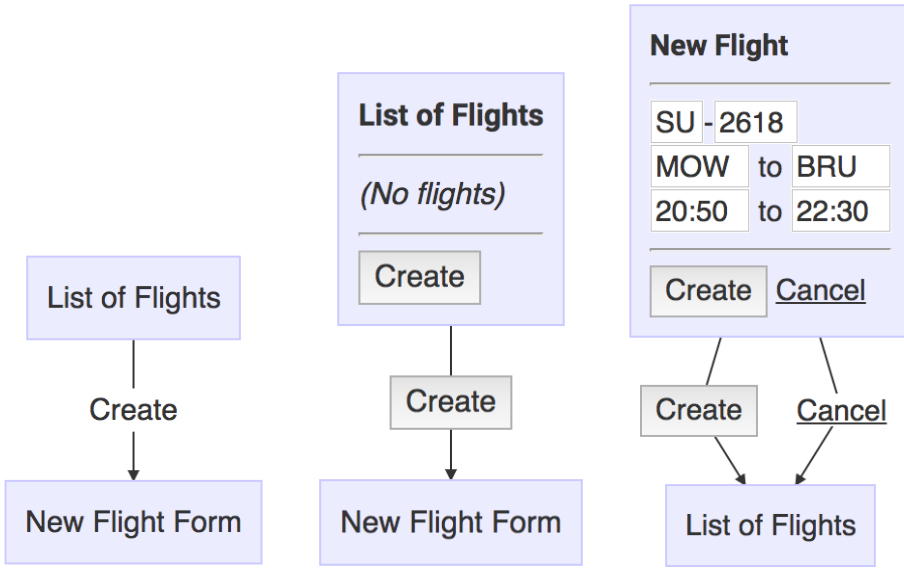


Функциональные макеты экранов (II)

```
graph TD
list["List of Flights"]
new["<b>New Flight</b><hr>
<input value='SU' size='2'>-
<input value='2618' size='4'><br>
<input value='MOW' size='5'> to
<input value='BRU' size='5'><br>
<input value='20:50' size='5'> to
<input value='22:30' size='5'>
<hr><button>Create</button>
<u>Cancel</u>"]
new-->|"
<button>Create</button>"|list
new-->|"<u>Cancel</u>"|list
```



Результаты


List of Flights
Create
New Flight Form
List of Flights
(No flights)
Create
Create
New Flight Form
New Flight
SU - 2618
MOW to BRU
20:50 to 22:30
Create Cancel
Create
Cancel
List of Flights

Без шаблонов тяжко: list

```
<b>${title List of Flights}</b><hr>

<i>(No flights)</i><hr>

${link new <button>Create</button>}
```

Без шаблонов тяжко: new

```
<b>${title New Flight}</b><hr>

<input value='SU' size='2'>-
<input value='2618' size='4'><br>

<input value='MOW' size='4'> to
<input value='BRU' size='4'><br>

<input value='20:50' size='5'> to
<input value='10:30' size='5'><hr>

${link list <button>Create</button>}
${link list <button>Cancel</button>}
```

Базовые хелперы

```
${title New Flight}
```

$title <text:*>

```
${link list <button>Create</button>}
```

$link <screen:word> <body:*>

Базовые хелперы: pass-through макросы

```
${define title {{'*', 'text'}} [[${text}]]}

${title New Flight} --> New Flight

${define link
  {{'word', 'target'}, {'*', 'body'}}
  [[${body}]]}

${link list <button>Create</button>}
--> <button>Create</button>
```

$define <symbol:word> <arguments:table> <code:*>

define

```
define = function(context, str)
  local symbol, str = eat.word(str)
  local args, str = eat.table(str)
  local code, str = eat['*'](str)
  args, code = lua_value(args), lua_value(code)
  if type(code) == 'string' then code =
    function(ctx) return ctx:replace(code) end
  end
  context._ROOT[symbol] = function(parent, str)
    local ctx = { }
    for i = 1, #args do
      ctx[args[i][1]], str = eat[args[i][2]](str)
    end
    return code(parent:push(ctx))
  end
end
```

Хелперы: код на Lua

```
${define title {{'*', 'text'}} function(context)
  local text = context:replace(context.text)
  context._ROOT._SCREENS[text] = text
  return text
end}
```

```
${define link {{'word', 'target'}, {'*', 'body'}}
function(context)
  local target = context:replace(context.text)
  context._ROOT._LINKS[target] = context._MODULE
  context:include(target) -- Ignoring result
  return context:replace(context.body)
end}
```

include

```
include = function(context, template)
  return context:push(
    { _MODULE = template }
  ):replace(
    assert(io.open(filename)):read("*a")
  )
end
```

Виды диаграмм

В зависимости от того, как определить $title and $link, можно получить несколько видов диаграмм из одного набора шаблонов:

- Outline (только заголовки и стрелки, ”переходы между экранами”)
- Closeup (содержимое ”текущего” экрана и переходы из него в другие экраны)
- Printable (только содержимое ”текущего” экрана)

Некоторые другие полезные хелперы

```
${define # {{'*', 'comment'}} [[]]}
${define --[HR]-------...---------- {} [[<hr>]]}

${define expr {{'*', 'code'}} function(context)
  return assert(loadstring('return '
    .. context:replace(context.code)))()
end}
```

Хелпер with

```
${define with {{'table', 'more_context'},
  {'*', 'body'}} function(context)
  return context:replace(
    context:push(context.more_context),
    context.body)
end}

${define form {} [[
  ${when editable
    <input value='MOW'> to <input value='BRU'>}
  ${unless editable MOW to BRU}
]]}

${with {editable = true} ${form}}
```

Хелпер with (II)

```
${define histogram {{'word', 'a'},
  {'word', 'b'}, {'word', 'c'}} [[
${with { w = '${expr ${a} + ${b} + {c}}' }
  <div style='width:${w}px' class='red'>
    <div style='width:${a}px'
      class='green'>${a}</div>
    <div style='width:${b}px'
      class='blue'>${b}</div>
  </div>
}]]}

${histogram 1 2 3}
```

Немного статистики

- Ядро примерно в 250 строк было написано за два дня.
- После шести месяцев нефуллтайм использования ядро выросло примерно до 330 строк кода. В основном была добавлена диагностика.
- Разработано порядка 60 макетов (5 тысяч строк кода шаблонов), будет ещё больше.

Стоит ли овчинка выделки?

Да.

- Получился дешёвый легковесный и гибкий фреймворк для разработки функциональных макетов, с которым мне лично легко работать.
- Выдача этого фреймворка, хотя и не идеальна, вполне доступна для понимания всеми членами команды.
- (И было прикольно написать ещё один шаблонизатор!)

Почему не X?

- Разработка Lugram потребовала настолько мало усилий, что адаптация существующего инструмента под мои нужды (или даже просто его освоение), вероятно, стоили было бы столько же или больше.
- Но если вы знаете подходящий инструмент — делитесь!

Проблемы

- Диагностика ошибок и отладка. В данный момент почти полностью отсутствуют. Небольшими усилиями можно добиться значительных улучшений.
- Отладка отрисовки HTML. Сложно как в IE6. Очень сложно улучшить. Нужно следить за тем, чтобы HTML выходил максимально простым.
- Можно расширить выразительные возможности языка. Но имеющихся пока вполне достаточно.

Вопросы?

@agladysh

agladysh@gmail.com

https://github.com/tais-aero/lugram