bumble badoo

Практики автоматизации тестирования мобильных приложений

Дмитрий Макаренко & Віктар Караневіч

Арти

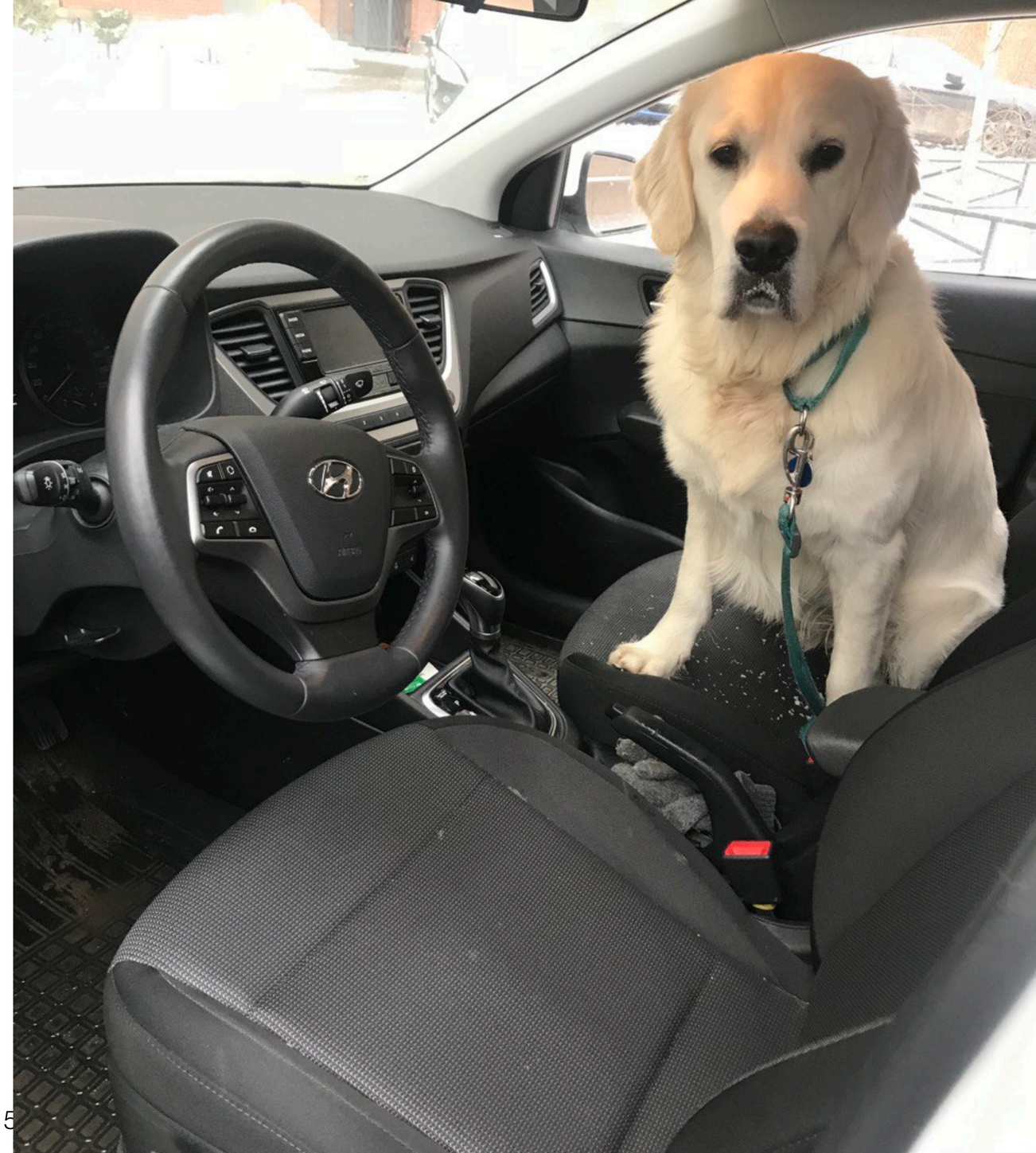
Slides expert

Дмитрий Макаренко

Mobile QA

5+ лет в тестировании

3 года назад переехал в Лондон



Тестирую приложения Badoo и Bumble

Віктар Караневіч

Senior Automation QA

Занимаюсь автоматизацией 10+ лет

Поддерживаю фреймворк iOS автоматизации, код ревью

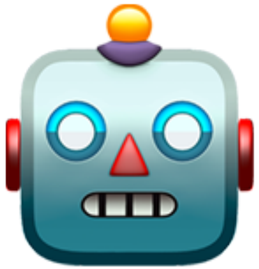
Bumble

Badoo

Процессы раньше

Команды мобильного тестирования

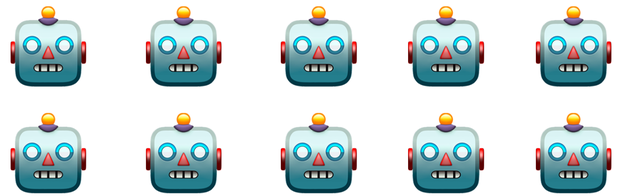
Команда автоматизации



Процессы раньше

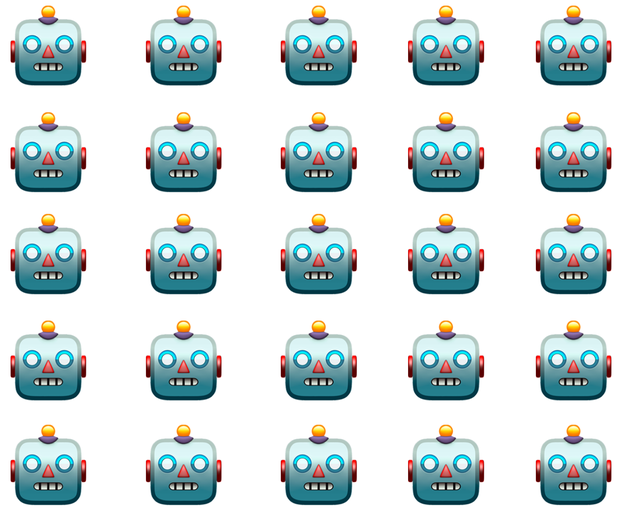
Команды мобильного тестирования

Команда автоматизации

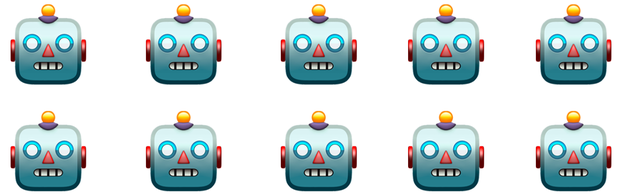


Процессы сейчас

Команды мобильного тестирования



Команда автоматизации



Требования к автоматизации

- Высокая скорость разработки
- Высокая стабильность тестов

Вам будет интересно, если вы

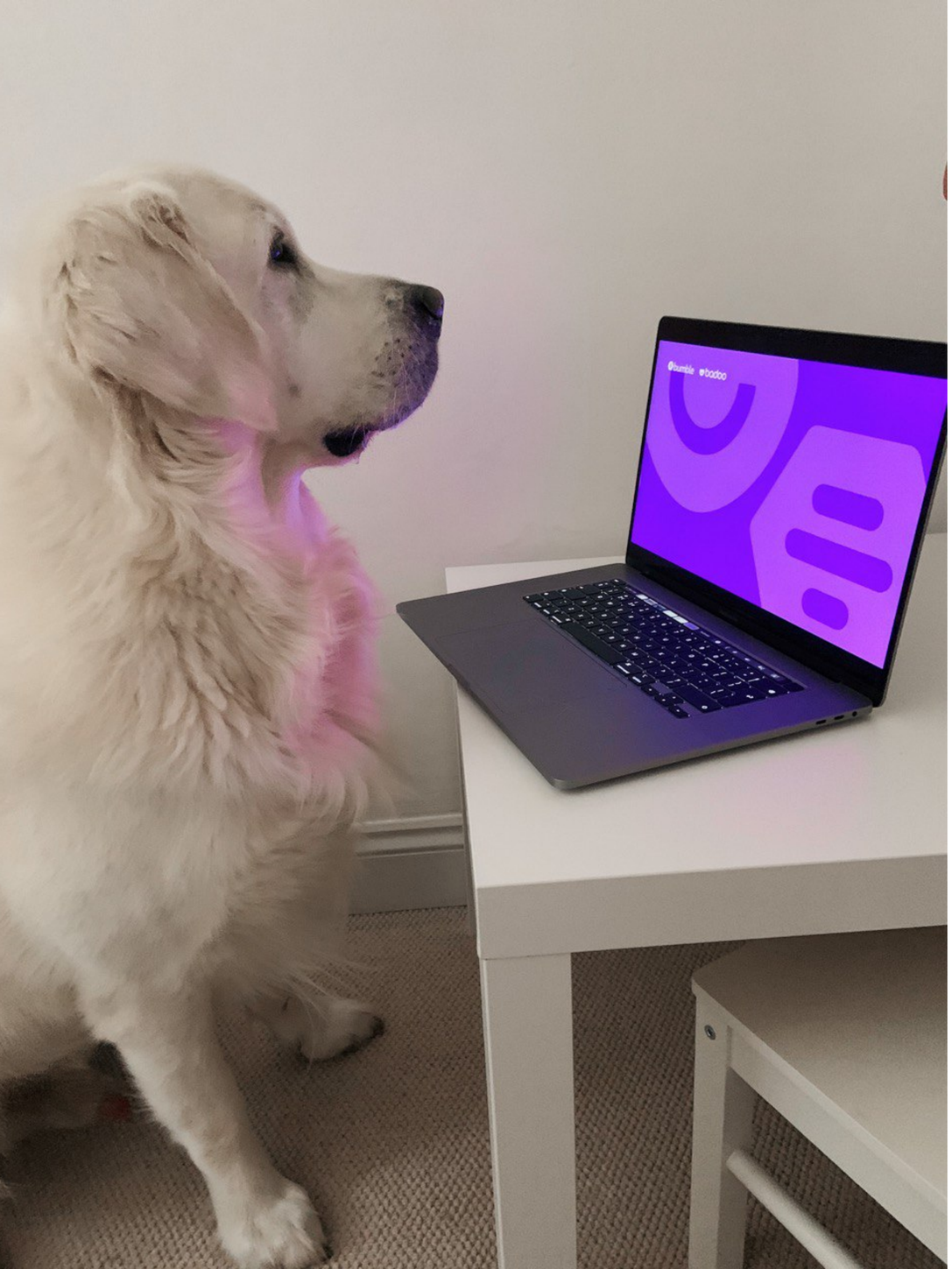
- Планируете внедрять автоматизацию
- Недавно начали это делать
- Активно занимаетесь автоматизацией

Выбор фреймворка

Выбор фреймворка

- **Переиспользование сценариев между iOS и Android**

Где писать проверки?



Или

Структура теста

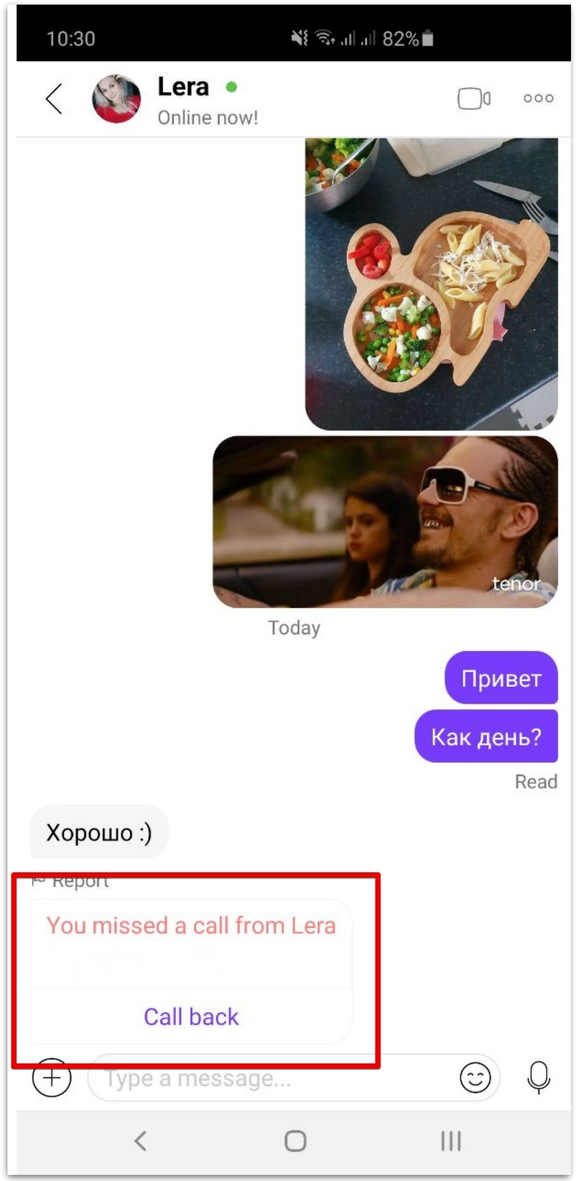
Scenario → Step Definitions → Page

Задача

- Создать проверку какого-либо элемента

Пример

- **Сообщение о пропущенном видеозвонке**



QaApi

API для изменения внутреннего состояния системы в процессе тестирования приложения

- **Отправить сообщения тестовому пользователю**
- **Загрузить фото профиля**
- **To be continued…**

https://bit.ly/3nr0YFo

Пример сценария

```
Scenario: Missed Video Call message is displayed in Chat
  Given users with following parameters
    | role             | name |
    | primary_user     | Dima |
    | video_call_user  | Lera |
  And   primary_user logs in
  And   primary_user receives a missed Video Call message from video_call_user via QaApi
  When  primary_user opens Chat with video_call_user
  Then  primary_user should have missed Video Call message from "Lera"
```

Пример


10:30
82%
Lera
Online now!
tenor
Today
Привет
Как день?
Read
Хорошо :)
Report
You missed a call from Lera
Call back
Type a message...

Пример шага

```
Then(/^([^"]*) should have missed Video Call message from "(.+)"$/) do |role, name|
  with_role(role) do
    Pages::ChatPage.new.await.verify_missed_video_call(name)
  end
end
```

Пример страницы

```
def verify_missed_video_call(name)
  expected = CallLexemes.missed_video_call_message(name)
  actual = ui.element_text(VIDEO_CALL_MESSAGE)
  Assertions.assert_equal(expected, actual,
                          "Missed Video Call message is incorrect")
end
```

Пример страницы

```
def verify_missed_video_call(name)
  expected = CallExemes.missed_video_call_message(name)
  actual = ui.element_text(VIDEO_CALL_MESSAGE)
  Assertions.assert_equal(expected, actual,
                          "Missed Video Call message is incorrect")
end
```

Пример страницы

```
def verify_missed_video_call(name)
  expected = CallLexemes.missed_video_call_message(name)
  Actual = ui.element_text(VIDEO_CALL_MESSAGE)
  Assertions.assert_equal(expected, actual,
                          "Missed Video Call message is incorrect")
end
```

Пример страницы. Другая платформа

```
def verify_missed_video_call(name)
  expected = CallLexemes.missed_video_call_message(name)
  actual = ui.element_text(VIDEO_CALL_MESSAGE)
  Assertions.assert_equal(expected, actual,
                          "Missed Video Call message is incorrect")

  expected = CallLexemes::CALL_BACK_BUTTON
  actual =  ui.element_text(CALL_BACK_BUTTON)
  Assertions.assert_equal(expected, actual,
                          "Call back button text is incorrect")
end
```

Проблемы

- Риск ошибки -> забытые/неправильно реализованные проверки
- Дублирование кода -> медленнее разработка

Пример шага. Исправленный вариант

```
Then(/^([^"]*) should have missed Video Call message from "(.+)"$/) do |role, name|
  with_role(role) do
    page = Pages::ChatPage.new.await

    expected = CallLexemes.missed_video_call_message(name)
    actual = page.video_call_message_text
    Assertions.assert_equal(expected, actual,
                            "Missed Video Call message is incorrect")

    expected = CallLexemes::CALL_BACK_BUTTON
    actual =  page.call_back_button_text
    Assertions.assert_equal(expected, actual,
                            "Call back button text is incorrect")
  end
end
```

Пример страницы. Исправленный вариант

```
def video_call_message_text
  ui.element_text(VIDEO_CALL_MESSAGE)
end

def call_back_button_text
  ui.element_text(CALL_BACK_BUTTON)
end
```

Выводы

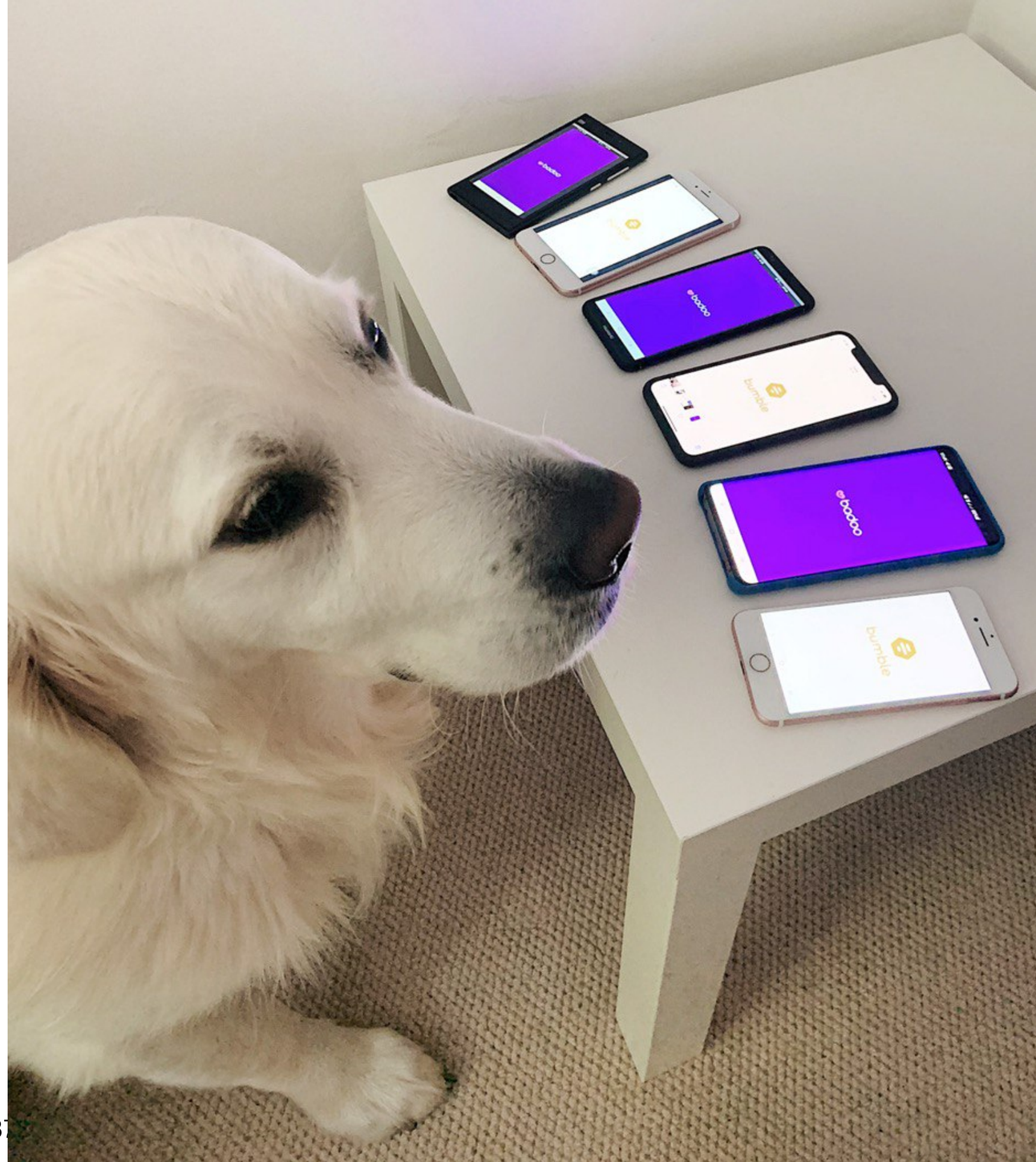
- Создаем проверки на уровне шагов
- Создаем страницы простыми

Применение

- Создавайте проверки на высоком уровне
- Создавайте объект тестирования простым

Работа с несколькими приложениями

“Несколько приложений? Легко!”

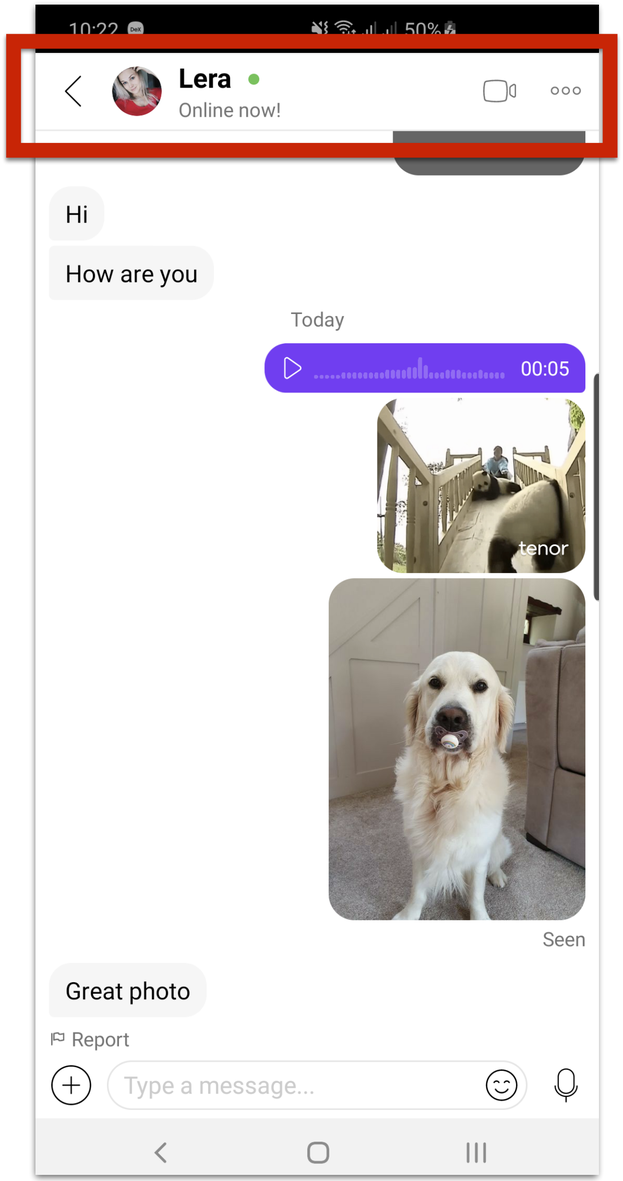


Задача

- **Переиспользовать страницу чата между Badoo и Bumble**

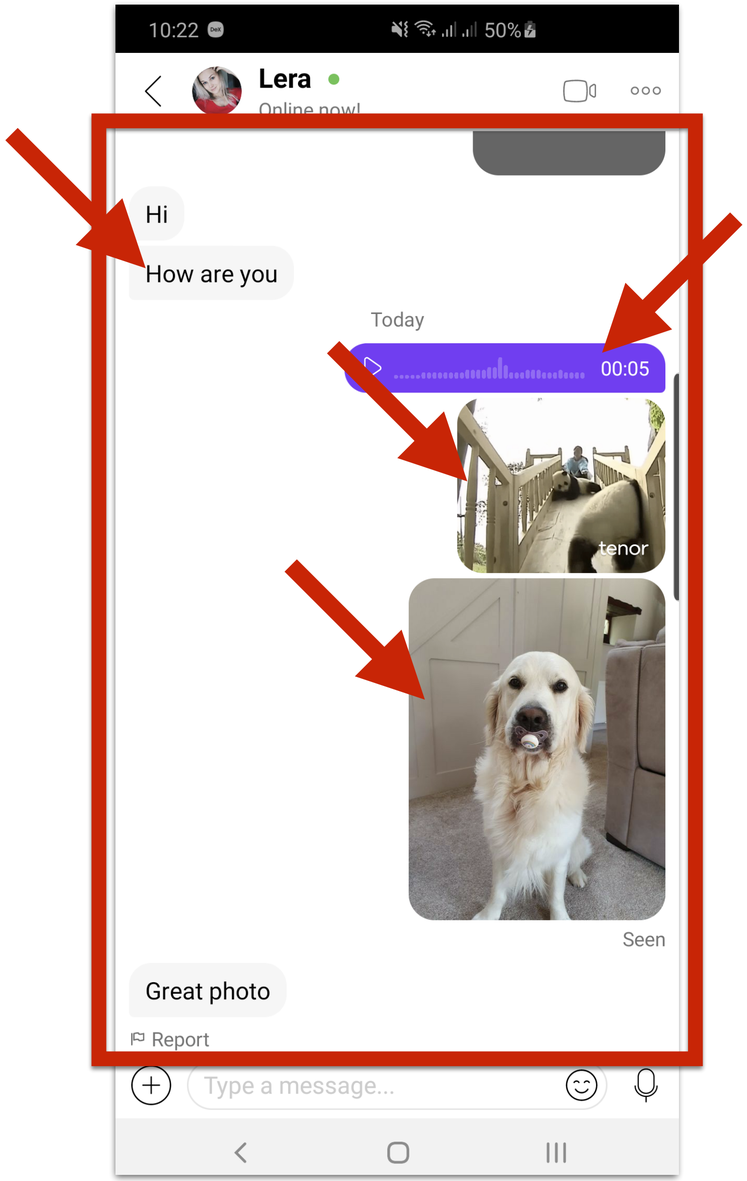
Создание модулей

Toolbar



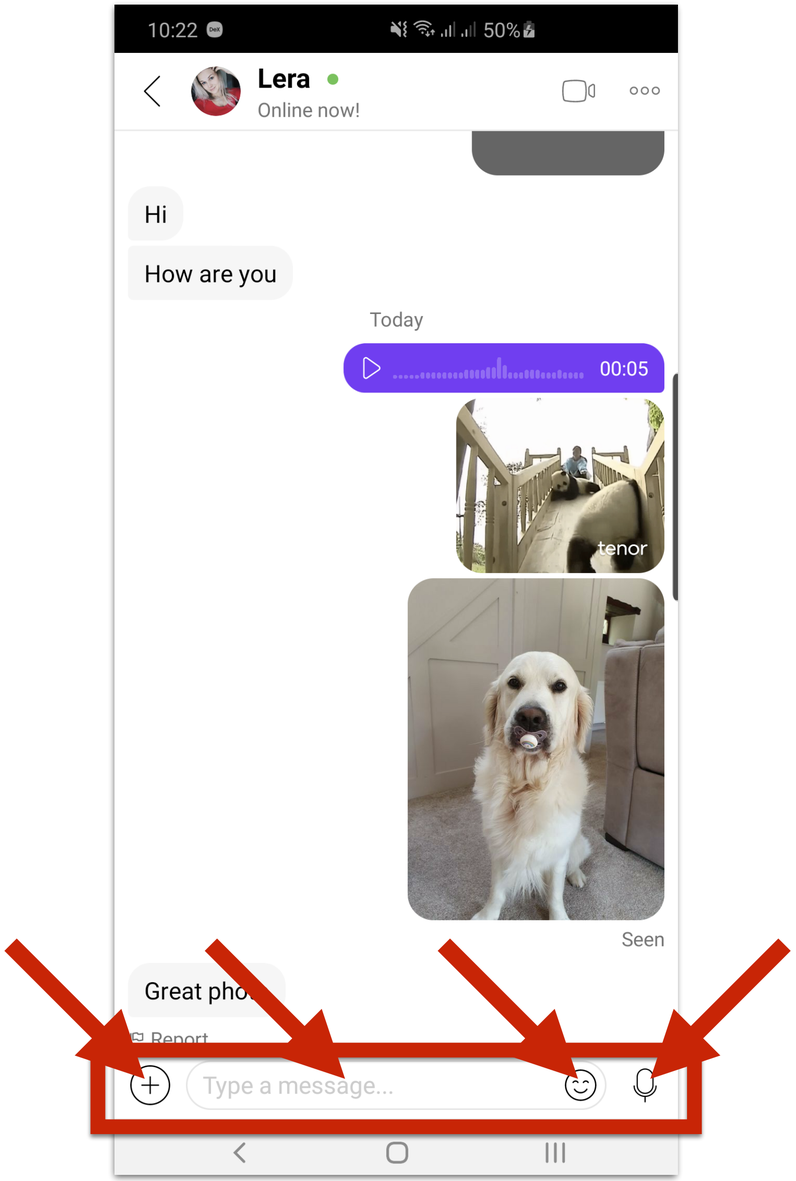
Создание модулей

Conversation

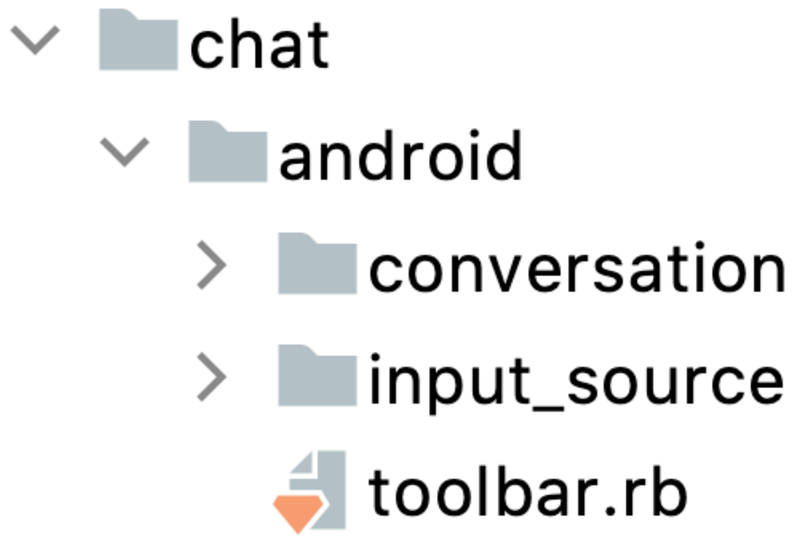


Создание модулей

Input source



Создание модулей



Использование общего компонента

```
class ChatPage < AndroidBadooBase
```

VS

```
class ChatPage < AndroidBumbleBase
```

Использование общего компонента

```
module Pages
  class ChatPage < AndroidBadooBase
    include Chat::Android::Conversation::Message::Text
    include Chat::Android::Conversation::Message::Photo
    include Chat::Android::Conversation::Message::Gif
    include Chat::Android::Conversation::Message::Audio
    include Chat::Android::Conversation::Message::Video
    include Chat::Android::Conversation::Message::VideoCall
    include Chat::Android::Conversation::Message::Gift
```

Использование общего компонента

```
module Pages
  class ChatPage < AndroidBumbleBase
    include Chat::Android::Conversation::Message::Text
    include Chat::Android::Conversation::Message::Photo
    include Chat::Android::Conversation::Message::Gif
    include Chat::Android::Conversation::Message::Audio
    include Chat::Android::Conversation::Message::Video
    include Chat::Android::Conversation::Message::VideoCall
    include Chat::Android::Conversation::Message::Reaction
```

Использование общего компонента

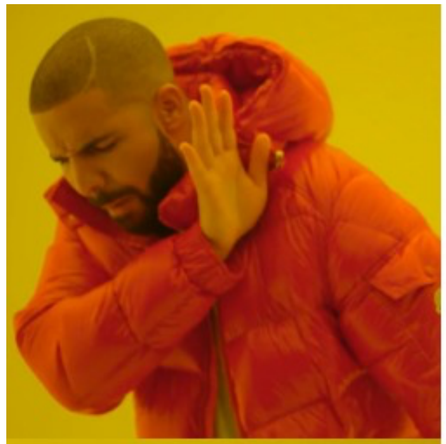
```
module Chat
  module Android
    module Conversation
      module Message
        module VideoCall
          VIDEO_CALL_MESSAGE = Locator.builder.id('message_videoChatText').build
          CALL_BACK_BUTTON   = Locator.builder.id('message_recall_button').build

          def video_call_message_text
            ui.element_text(VIDEO_CALL_MESSAGE)
          end

          def call_back_button_text
            ui.element_text(CALL_BACK)
          end
        end
      end
    end
  end
end
```

Выводы

- Пишем новые тесты для разных продуктов, комбинируя существующие шаги
- Используем компоненты для страниц с большим количеством UI-элементов

Применение

- **Создавайте компоненты для объектов с большим количеством свойств/методов**



God
object

Разбить
на модули

Шаги для базовых действий

Базовые действия

- Await
- Verify
- Close
- Go back

Задача

- Реализовать базовые действия для разных страниц

Пример ожидания страниц

```
Then(/^primary_user waits for Chat Page$/) do
  Pages::ChatPage.new.await
end

Then(/^primary_user is on Own Profile page$/) do
  Pages::OwnProfile.new.await
End
```

Пример возвращения со страниц

```
When(/^primary_user taps back on Connections Page$/) do
  Pages::ConnectionsPage.new.await.tap_back
end

When(/^primary_user returns from Encounters page$/) do
  Pages::EncountersPage.new.await.go_back
end

When(/^primary_user goes back from Chat Page$/) do
  Pages::ChatPage.new.await.press_back
end
```

Проблема

- Дублирование кода -> медленнее разработка
  - Определения шагов
  - Реализация методов на страницах

Пример шагов для базовых действий

```
Then(/^primary_user waits for (.*) [P|p]age$/) do |page_name|
  page_object_by(page_name).await
end

When(/^primary_user goes back from (.*) [P|p]age$/) do |page_name|
  page_object_by(page_name).await.press_back
end

Then(/^primary_user verifies (.*) [P|p]age$/) do |page_name|
  page_object_by(page_name).await.verify_default_screen
end
```

Пример шагов для базовых действий

```
Then(/^primary_user waits for (.*) [P|p]age$/) do |page_name|
  page_object_by(page_name).await
end

When(/^primary_user goes back from (.*) [P|p]age$/) do |page_name|
  page_object_by(page_name).await.press_back
end

Then(/^primary_user verifies (.*) [P|p]age$/) do |page_name|
  page_object_by(page_name).await.verify_default_screen
end
```

Преобразование страниц

```
def page_object_by(page_name)
  page_class = "#{page_name} page".downcase.split(' ').collect(&:capitalize).join

  return Pages.const_get(page_class).new if Pages.const_defined?(page_class)

  raise(NameError, "Unknown class #{page_class}. Actual page name: '#{page_name}'")
end
```

Преобразование страниц

```
def page_object_by(page_name)
  page_class = "#{page_name} page".downcase.split(' ').collect(&:capitalize).join

  return Pages.const_get(page_class).new if Pages.const_defined?(page_class)

  raise(NameError, "Unknown class #{page_class}. Actual page name: '#{page_name}'")
end
```

Преобразование страниц

```
And    primary_user waits for Chat page
```

page_object_by

```
ChatPage
```

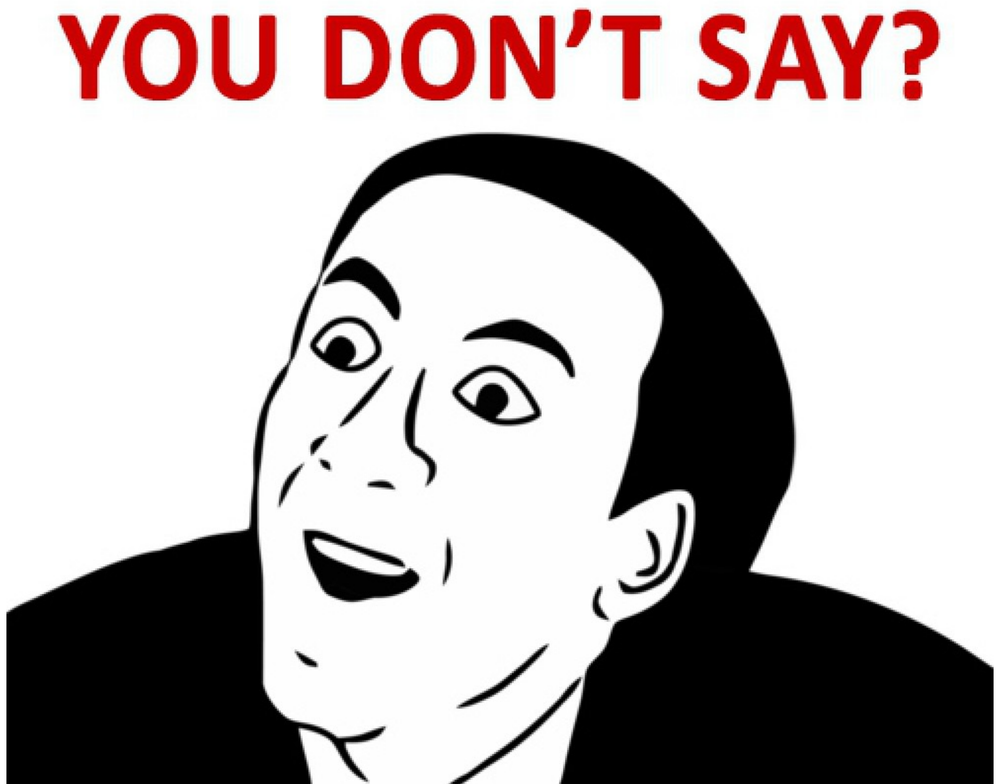
```
Pages::ChatPage.new.await
```

Вывод

- Создаем общие шаги для базовых действий
- Создаем общие методы на страницах

Применение

- Если у вас много однотипного кода, выделяйте общие методы



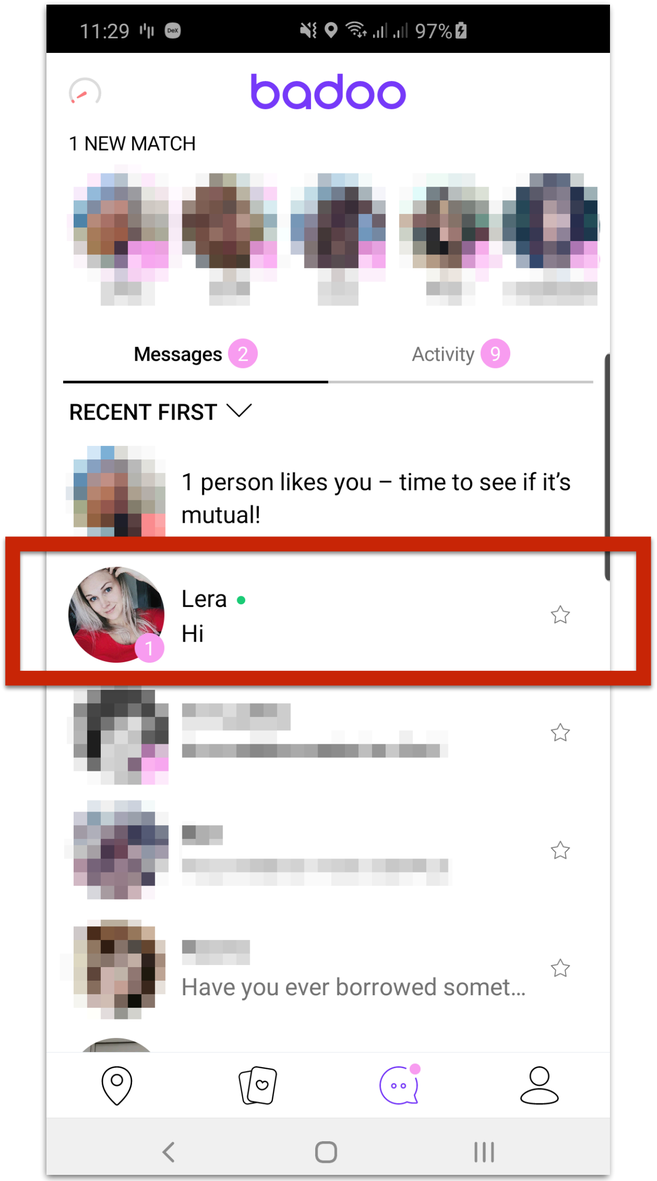
Верификация изменения состояний элементов

Динамическая среда

- Загрузка элементов

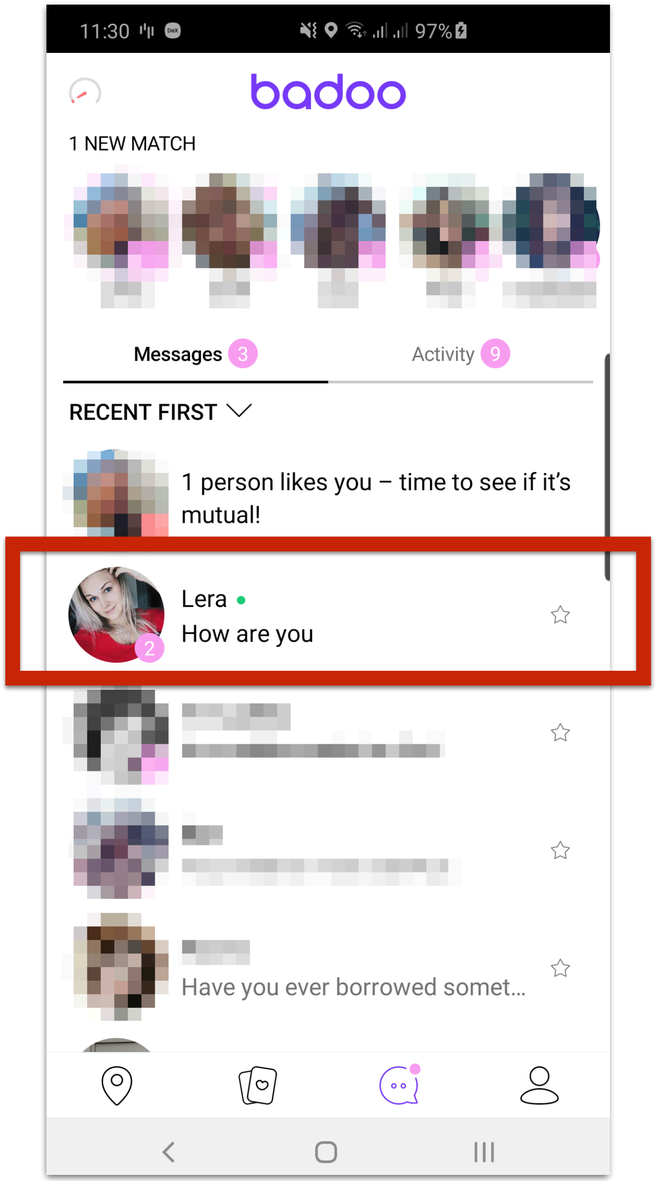
Динамическая среда

- Загрузка элементов
- **Изменение состояний элементов**



Динамическая среда

- Загрузка элементов
- **Изменение состояний элементов**



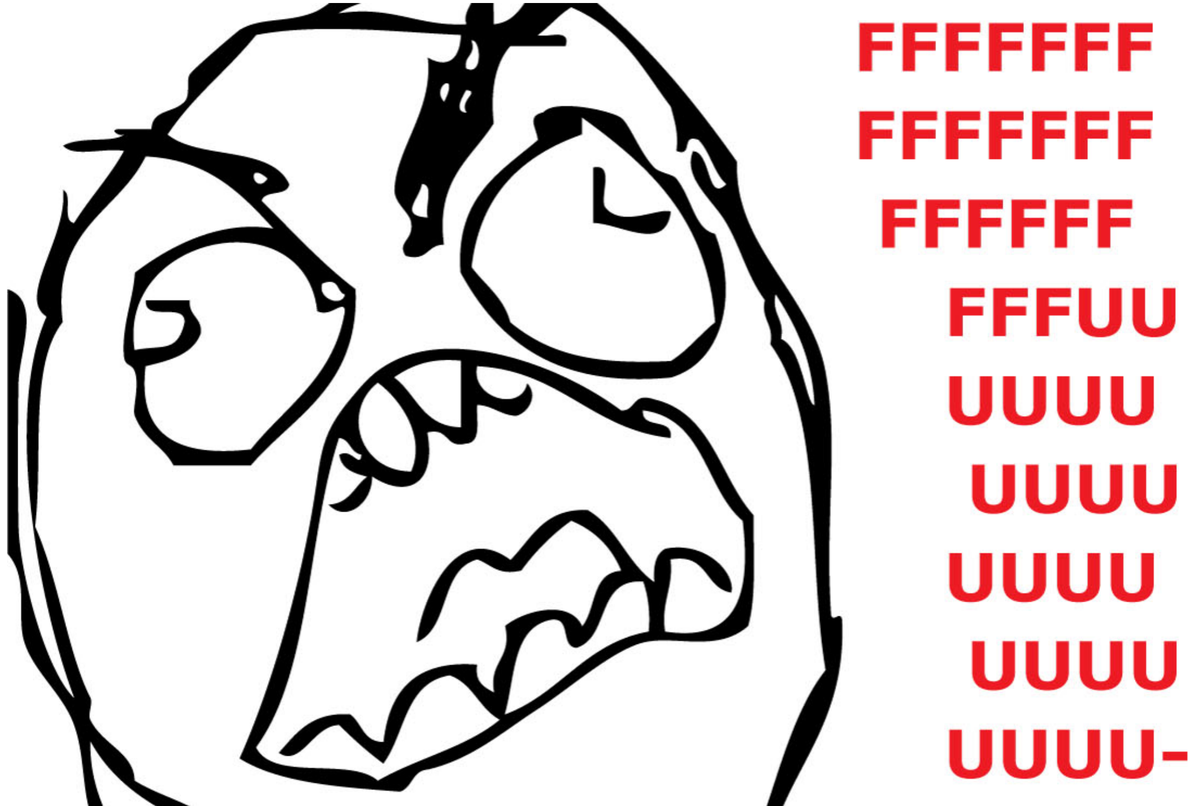
Стандартные решения

- Selenium - Implicit/Explicit/Fluent waits
- Calabash - wait_for

Yukihiro Matsumoto
“Matz”

Стандартные решения. Проблема

- wait_for использует Ruby Timeout
- Может бесконечно зависать



Стандартные решения. Проблема

```
def scroll_to_block_button
  wait_for(timeout: 30) do
    scroll_down
    wait_until_no_animation
    ui.element_displayed?(BLOCK_BUTTON)
  end
end
```

Стандартные решения. Проблема

```
def scroll_to_block_button
  wait_for(timeout: 30) do
    scroll_down
    wait_until_no_animation
    ui.element_displayed?(BLOCK_BUTTON)
  end
end
```

```
def wait_until_no_animation
  wait_for(timeout: 10) do
    !ui.any_element_animating?
  end
end
```

Стандартные решения. Проблема

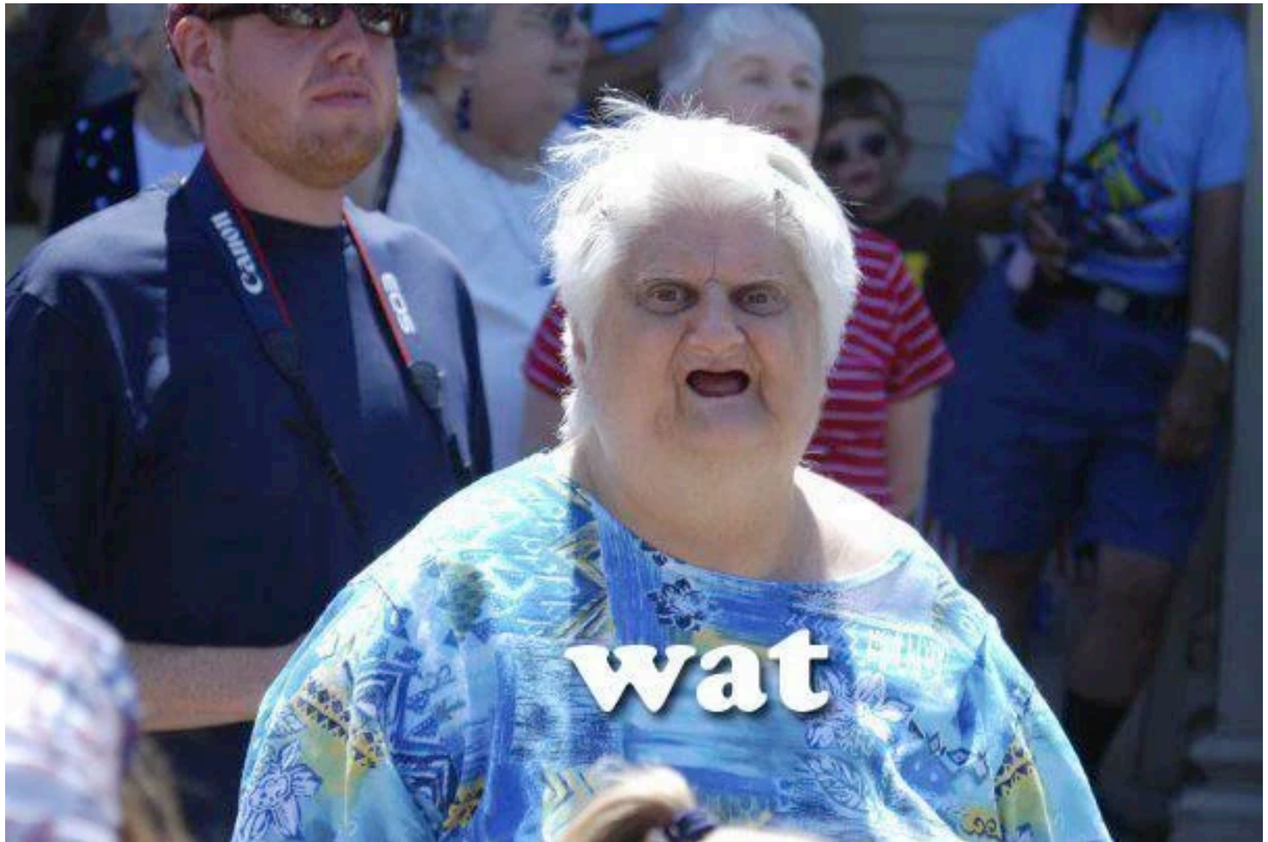
```
def scroll_to_block_button
  wait_for(timeout: 30) do
    scroll_down
    wait_until_no_animation
    ui.element_displayed?(BLOCK_BUTTON)
  end
end


def wait_until_no_animation
  wait_for(timeout: 10) do
    !ui.any_element_animating?
  end
end
```

Свой Poll модуль. Требования

- Проверка повторяется, пока не выполнится заданное условие
- Ошибка, если условие не выполнилось в течение заданного времени

Свой Poll модуль. return_on_timeout

- Возвращает результат вместо ошибки по таймауту



Свой Poll модуль. return_on_timeout

```
Poll.for { 2 > 3 }
> WaitError
```

Свой Poll модуль. return_on_timeout

```
Poll.for(return_on_timeout: true) { 2 > 3 }
> false
```

Свой Poll модуль. return_on_timeout

- Используется для верификации изменения состояний элементов

Может скрывать настоящие падения


return_on_timeout: true

Варианты изменения состояний

Варианты изменения состояний

- Должен появиться


QUIZ
TIME

Сколько всего вариантов изменения состояний?

Варианты изменения состояний

- Должен появиться
- Должен пропасть

1

Варианты изменения состояний

- Должен появиться
- Должен пропасть

Варианты изменения состояний

- Должен появиться
- Должен пропасть
- Не должен появиться

Варианты изменения состояний

- Должен появиться
- Должен пропасть
- Не должен появиться
- Не должен пропасть

Реализация проверок

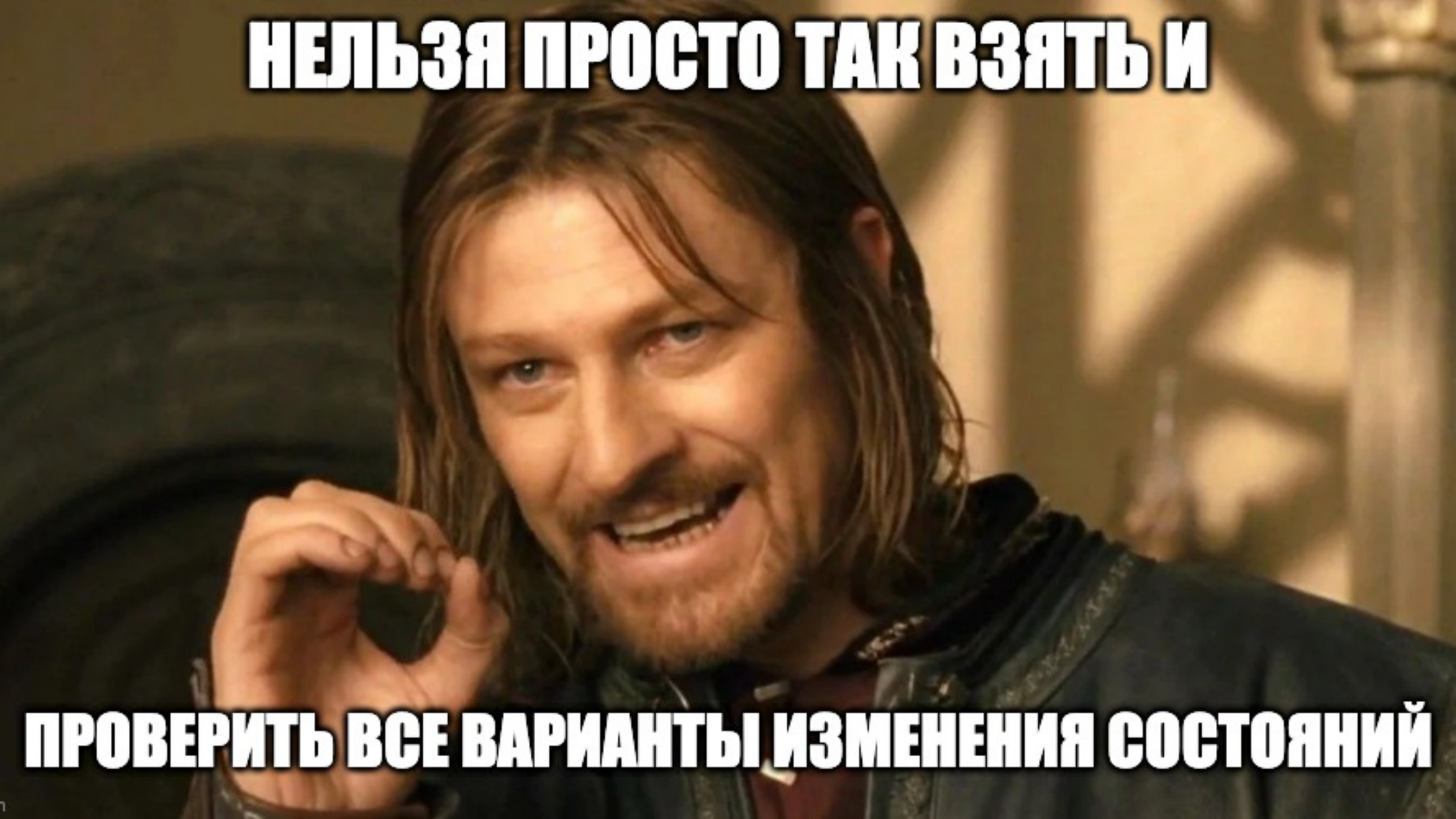
- Должен появиться
- Должен пропасть

Реализация проверок

```
# вариант "Должен появиться"
Poll.for_true { ui.elements_displayed?(locator) }


# вариант "Должен пропасть”
Poll.for_false { ui.elements_displayed?(locator) }
```

Реализация проверок

- Не должен появиться
- Не должен пропасть


НЕЛЬЗЯ ПРОСТО ТАК ВЗЯТЬ И
ПРОВЕРИТЬ ВСЕ ВАРИАНТЫ ИЗМЕНЕНИЯ СОСТОЯНИЙ

Реализация проверок

```
# вариант “Не должен пропасть”
sleep 10
Poll.for_true { ui.elements_displayed?(locator) }
```

Реализация проверок

- Увеличивается время тестов
- Можно пропустить баг, когда элемент появится и пропадет во время sleep

Реализация проверок

```
# вариант “Не должен появиться”
ui.wait_for_elements_not_displayed(locator)
actual_state = Poll.for(return_on_timeout: true) { ui.elements_displayed?(locator) }
Assertions.assert_false(actual_state, "Element #{locator} should not appear")
```

Реализация проверок

```
# вариант “Не должен появиться”
ui.wait_for_elements_not_displayed(locator)
actual_state = Poll.for(return_on_timeout: true) { ui.elements_displayed?(locator) }
Assertions.assert_false(actual_state, "Element #{locator} should not appear")
```

Реализация проверок

```
# вариант “Не должен появиться”
ui.wait_for_elements_not_displayed(locator)
actual_state = Poll.for(return_on_timeout: true) { ui.elements_displayed?(locator) }
Assertions.assert_false(actual_state, "Element #{locator} should not appear")
```

Реализация проверок

```
# вариант “Не должен появиться”
ui.wait_for_elements_not_displayed(locator)
actual_state = Poll.for(return_on_timeout: true) { ui.elements_displayed?(locator) }
Assertions.assert_false(actual_state, "Element #{locator} should not appear")
```

```
# вариант “Не должен пропасть”
ui.wait_for_elements_displayed(locator)
actual_state = Poll.for(return_on_timeout: true) { !ui.elements_displayed?(locator) }
Assertions.assert_false(actual_state, "Element #{locator} should not disappear")
```

Общий метод для всех вариантов

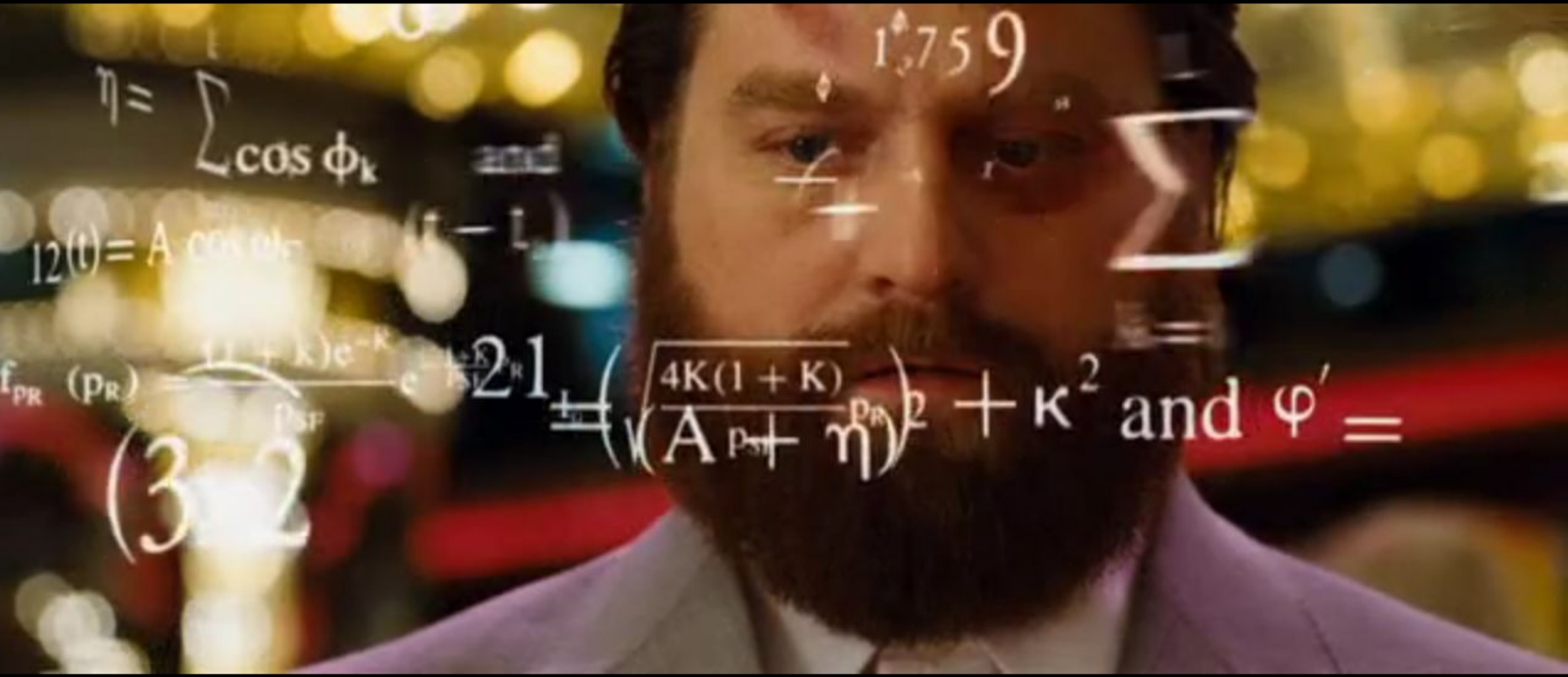
```
def verify_dynamic_state(state:, timeout: 10, error_message:)
  options = {
    return_on_timeout: true,
    timeout:           timeout,
  }

  case state
    when 'should appear'
      actual_state = Poll.for(options) { yield }
      Assertions.assert_true(actual_state, error_message)
    when 'should disappear'
      actual_state = Poll.for(options) { !yield }
      Assertions.assert_true(actual_state, error_message)
    when 'should not appear'
      actual_state = Poll.for(options) { yield }
      Assertions.assert_false(actual_state, error_message)
    when 'should not disappear'
      actual_state = Poll.for(options) { !yield }
      Assertions.assert_false(actual_state, error_message)
    else
      raise("Undefined state: #{state}")
  end
end
```

Вывод

- Важно обрабатывать все 4 варианта изменения состояний
- Полезно создать общий метод

Применение

- **Используйте полную систему проверок для вариантов изменения состояний**


$\eta = \sum \cos \phi_k$
$12(t) = A \cos(\omega t -$
$f_{PR}(p_R)$
$\pm \sqrt{\left(\frac{4K(1+K)}{A p_{sr}} + \eta\right) p_R^2 + \kappa^2}$ and $\varphi' =$

Применение

- **Используйте полную систему проверок для вариантов изменения состояний**

Должен пропасть	TRUE	FALSE
Не должен пропасть	TRUE	TRUE
Не должен появиться	FALSE	FALSE
Должен появиться	FALSE	TRUE

Надежная настройка предусловий теста

D

Задача

- **Настроить предусловия перед началом выполнения теста**

Примеры

- Отключение сервиса локации
- Создание истории чата

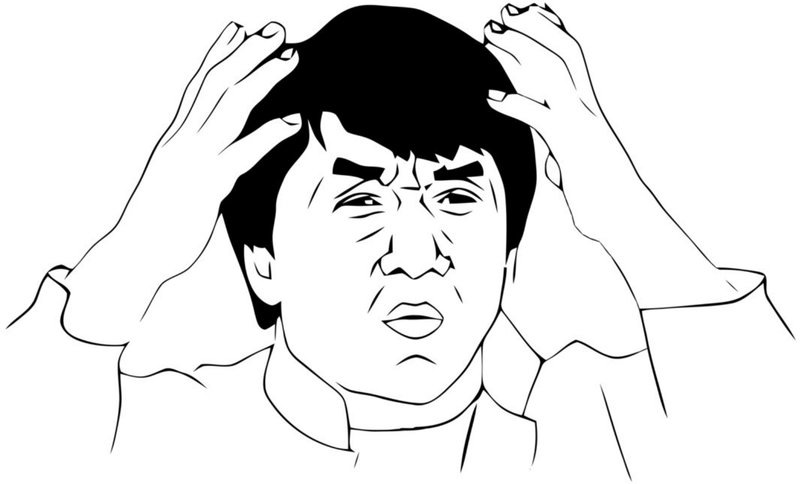
Отлючение сервиса локации

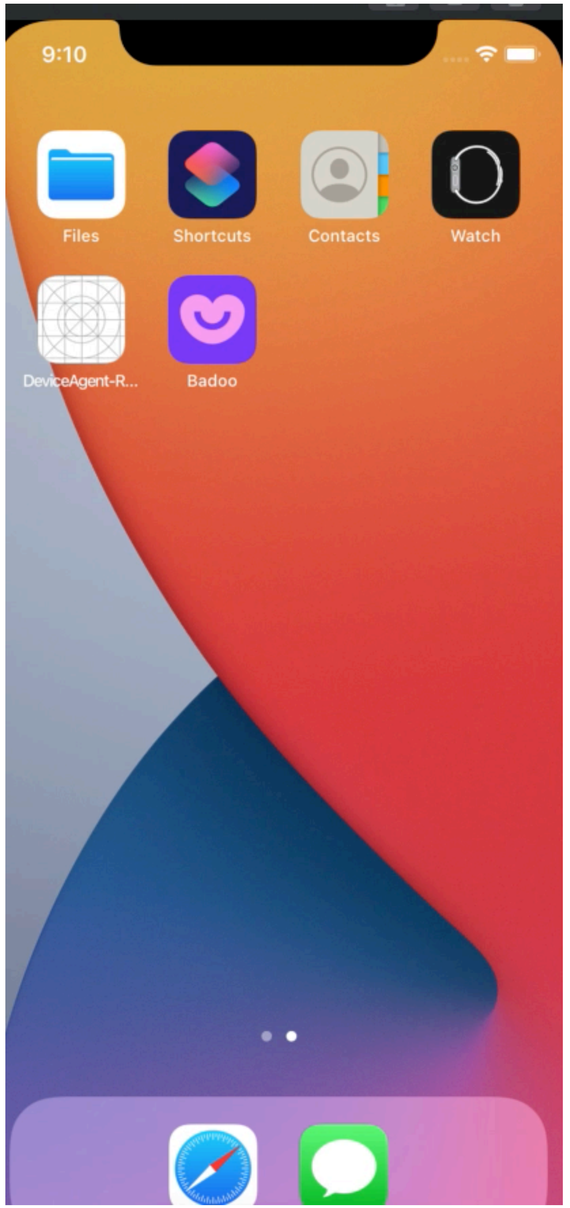
```
def switch off location service
  ui.wait_for_elements_displayed(SWITCH)

  if ui.element_value(SWITCH) == ON
    ui.tap_element(SWITCH)
    ui.tap_element(TURN_OFF)
  end
end
```

Отлючение сервиса локации

```
def switch_off_location_service
  ui.wait_for_elements_displayed(SWITCH)

  if ui.element_value(SWITCH) == ON
    ui.tap_element(SWITCH)
    ui.tap_element(TURN_OFF)
  end
end
```

Проблема





При быстром выходе из Настроек переключение может не сработать

Проверка сообщений в чате

- Проверяем загрузку истории в чате

Отправка сообщений в чате

```
def send_message(from:, to:, message:, count:)
  count.times do
    QaApi.chat_send_message(user_id: from, contact_user_id: to, message: message)
  end
end
```

Проблема

- При задержке на сервере сообщения могут не быть доставлены вовремя

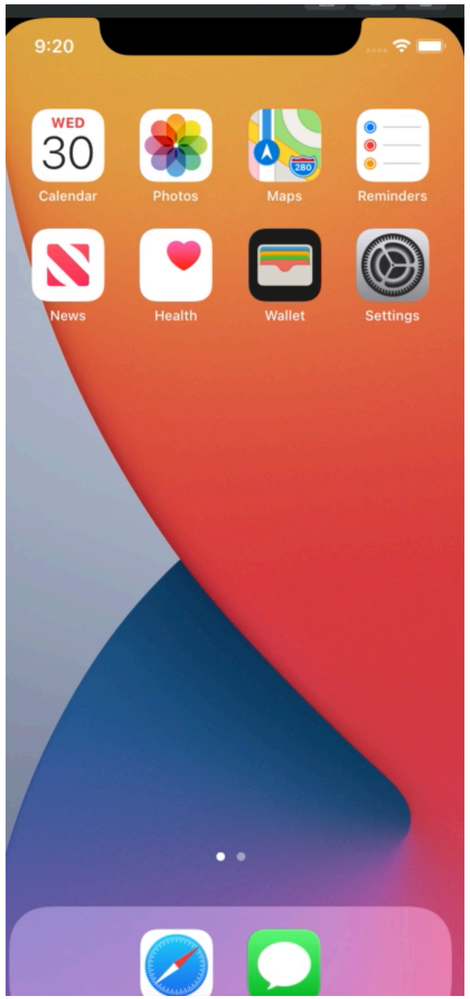
Общая проблема настройки предусловий

- Тесты могут падать в случае, когда приложение работает правильно

Надежное отключение сервиса локации

```
def ensure_location_services_switch_in_state_off
  ui.wait_for_elements_displayed(SWITCH)

  if ui.element_value(SWITCH) == ON
    ui.tap_element(SWITCH)
    ui.tap_element(TURN_OFF)

    Poll.for(timeout_message: 'Location Services should be disabled') do
      ui.element_value(SWITCH) == OFF
    end
  end
end
```

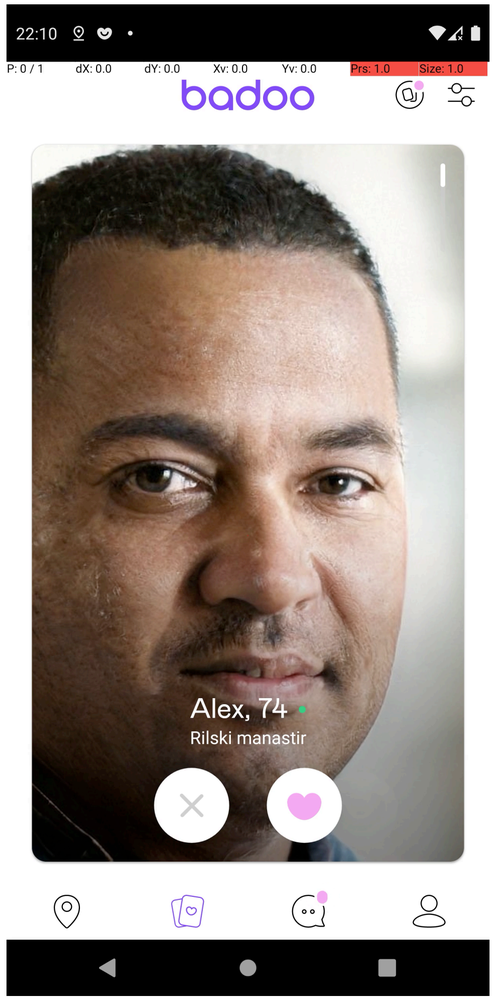
Надежное отключение сервиса локации



Надежная отправка сообщений в чате

```
def send_message(from:, to:, message:, count:)
  actual_messages_count = QaApi.received_messages_count(to, from)
  expected_messages_count = actual_messages_count + count

  count.times do
    QaApi.chat_send_message(user_id: from, contact_user_id: to, message: message)
  end

  QaApi.wait_for_user_received_message(from, to, expected_messages_count)
end
```

Проверка сообщения в чате



Вывод

- Необходимо дождаться установки предусловия перед продолжением теста
- Тесты должны падать при неуспешной установке предусловия

Применение

- **Добавляйте проверку предустановок, когда действия асинхронны**

https://martinfowler.com/articles/nonDeterminism.html

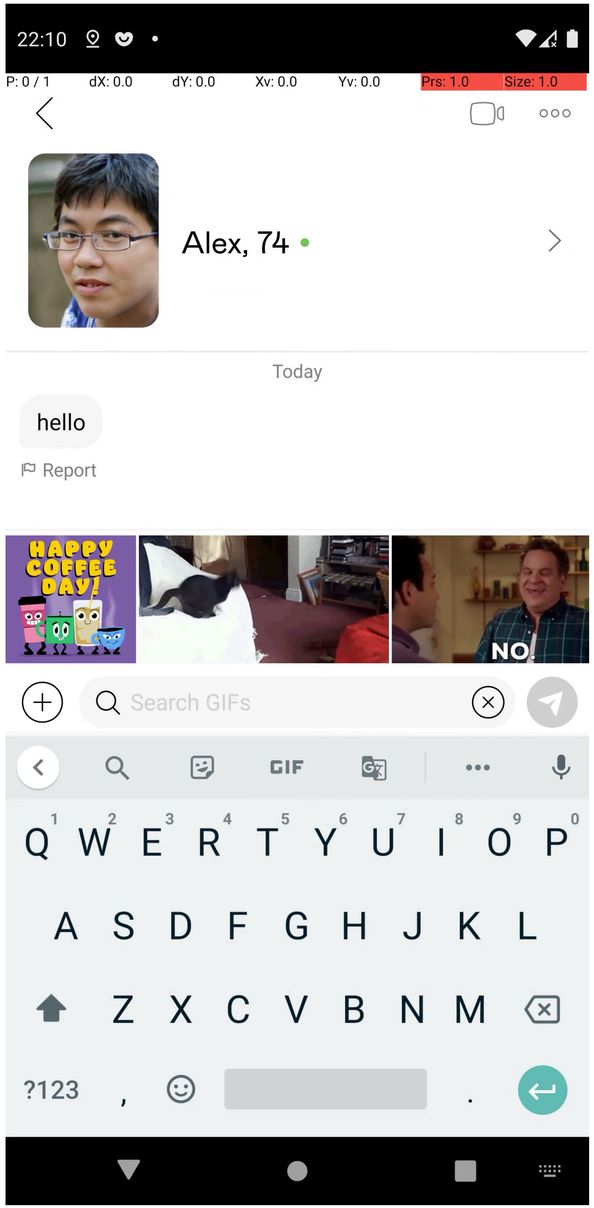
Независимость шагов

Задача

- **Реализовать шаги для разных действий/ верификаций**

Пример

- Поиск и отправка gif



Пример

```
Scenario: Searching and Sending GIF in Chat
  Given users with following parameters
    | role         | name |
    | primary_user | Dima |
    | chat_user    | Lera |
  And   primary_user logs in
  When  primary_user opens Chat with chat_user
  And   primary_user switches to GIF input source
  And   primary_user searches for "bee" GIFs
  And   primary_user sends 7th GIF in the list
  Then  primary_user verifies that the selected GIF has been sent
```

Пример

```
Scenario: Searching and Sending GIF in Chat
  Given users with following parameters
    | role         | name |
    | primary_user | Dima |
    | chat_user    | Lera |
  And   primary_user logs in
  When  primary_user opens Chat with chat_user
  And   primary_user switches to GIF input source
  And   primary_user searches for "bee" GIFs
  And   primary_user sends 7th GIF in the list
  Then  primary_user verifies that the selected GIF has been sent
```

Пример

```
Scenario: Searching and Sending GIF in Chat
  Given users with following parameters
    | role         | name |
    | primary_user | Dima |
    | chat_user    | Lera |
  And   primary_user logs in
  When  primary_user opens Chat with chat_user
  And   primary_user switches to GIF input source
  And   primary_user searches for "bee" GIFs
  And   primary_user sends 7th GIF in the list
  Then  primary_user verifies that the selected GIF has been sent
```

Пример

```
Scenario: Searching and Sending GIF in Chat
  Given users with following parameters
    | role           | name |
    | primary_user   | Dima |
    | chat_user      | Lera |
  And   primary_user logs in
  When  primary_user opens Chat with chat_user
  And   primary_user switches to GIF input source
  And   primary_user searches for "bee" GIFs
  And   primary_user sends 7th GIF in the list
  Then  primary_user verifies that the selected GIF has been sent
```

Пример

```
And(/^primary_user searches for "(.+)" GIFs$/) do |keyword|
  chat_page = Pages::ChatPage.new.await
  TestData.gif_list = chat_page.gif_list
  chat_page.search_for_gifs(keyword)
  Poll.for_true(timeout_message: 'Gif list is not updated') do
    (TestData.gif_list & chat_page.gif_list).empty?
  end
end
```

Пример

```
And(/^primary_user searches for "(.+)" GIFs$/) do |keyword|
  chat_page = Pages::ChatPage.new.await
  TestData.gif_list = chat_page.gif_list
  chat_page.search_for_gifs(keyword)
  Poll.for_true(timeout_message: 'Gif list is not updated') do
    (TestData.gif_list & chat_page.gif_list).empty?
  end
end
```

Пример

```
And(/^primary_user searches for "(.+)" GIFs$/) do |keyword|
  chat_page = Pages::ChatPage.new.await
  TestData.gif_list = chat_page.gif_list
  chat_page.search_for_gifs(keyword)
  Poll.for_true(timeout_message: 'Gif list is not updated') do
    (TestData.gif_list & chat_page.gif_list).empty?
  end
end
```

Пример

```
And(/^primary_user searches for "(.+)" GIFs$/) do |keyword|
  chat_page = Pages::ChatPage.new.await
  TestData.gif_list = chat_page.gif_list
  chat_page.search_for_gifs(keyword)
  Poll.for_true(timeout_message: 'Gif list is not updated') do
    (TestData.gif_list & chat_page.gif_list).empty?
  end
end
```

Проблема

- Сложно переиспользовать

Исправленный пример

```
Scenario: Searching and Sending GIF in Chat
  Given users with following parameters
    | role           | name |
    | primary_user   | Dima |
    | chat_user      | Lera |
  And   primary_user logs in
  When  primary_user opens Chat with chat_user
  And   primary_user switches to GIF input source
  And   primary_user stores the current list of GIFs
  And   primary_user searches for "bee" GIFs
  Then  primary_user verifies that list of GIFs is updated
  When  primary_user sends 7th GIF in the list
  Then  primary_user verifies that the selected GIF has been sent
```

Исправленный пример

```
And(/^primary_user stores the current list of GIFs$/) do
  TestData.gif_list = Pages::ChatPage.new.await.gif_list
end
```

Исправленный пример

```
And(/^primary_user searches for "(.+)" GIFs$/) do |keyword|
  Pages::ChatPage.new.await.search_for_gifs(keyword)
end
```

Исправленный пример

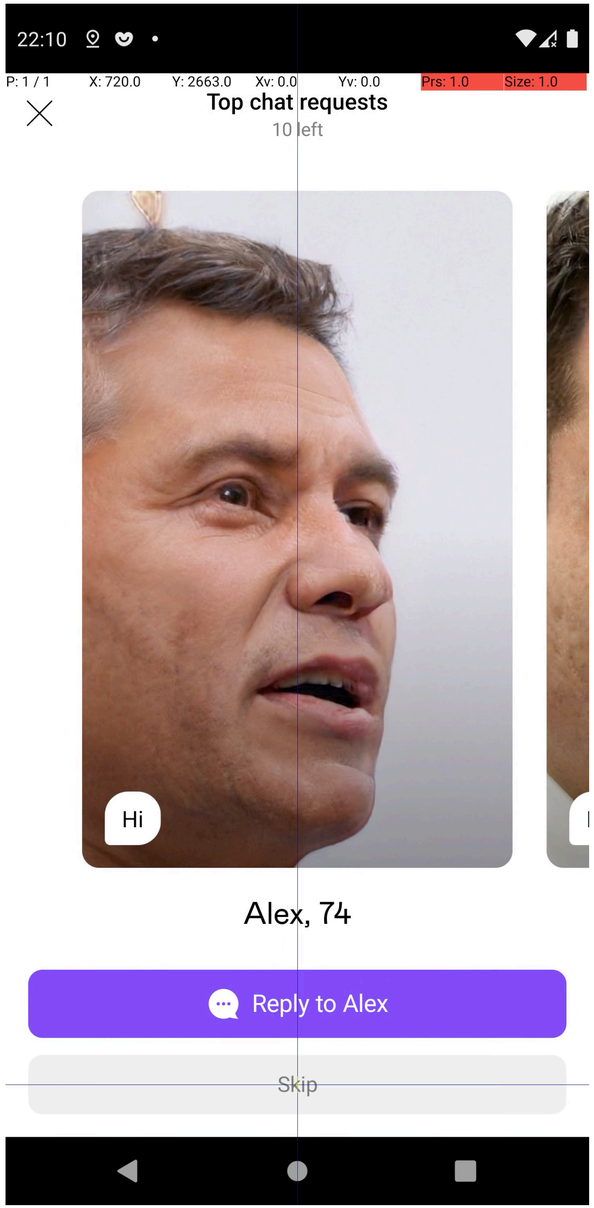
```
And(/^primary_user verifies that list of GIFs is updated$/) do
  chat_page = Pages::ChatPage.new.await
  Poll.for_true(timeout_message: 'Gif list is not updated') do
    (TestData.gif_list & chat_page.gif_list).empty?
  end
end
```

Исключения

- Шаги с переходами между экранами
- Шаги с изменением состояния одного экрана

Пример

- Голосование в мини-игре



Пример

```
When(/^primary_user votes No in Messenger mini game (\d+) times$/) do |count|
  page = Pages::MessengerMiniGamePage.new.await

  count.to_i.times do
    page.vote_no
  end
end
```

Проблема

- Голос может не успевать обработаться -> шаг не выполнит свою задачу

Исправленный пример

```
When(/^primary_user votes No with awaiting next profile in Messenger mini game (\d+) times$/) do |count|
  page = Pages::MessengerMiniGamePage.new.await

  count.to_i.times do
    progress_before = page.progress
    page.vote_no
    Poll.for_true do
      page.progress > progress_before
    end
  end
end
```

Вывод

- Создаем независимые шаги для простых действий
- В шагах для сложных действий добавляем дополнительные проверки

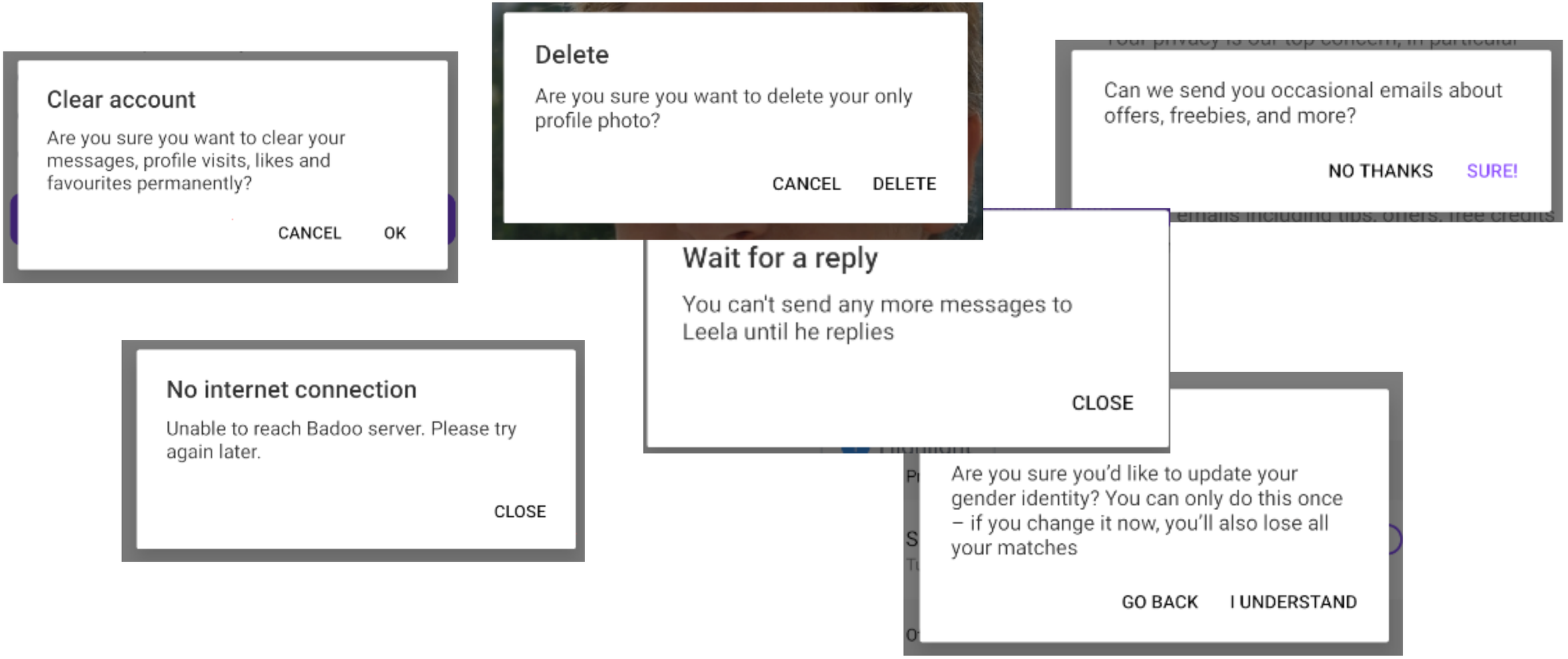
Применение

- Выделяйте независимые методы для простых действий в тестах
- Добавляйте проверку предустановок в сложных действиях

Верификация “необязательных” элементов

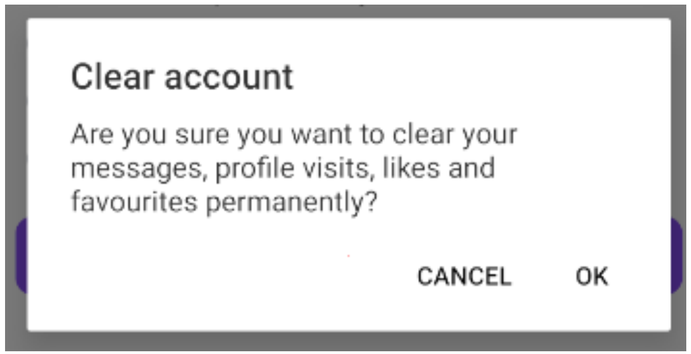
и

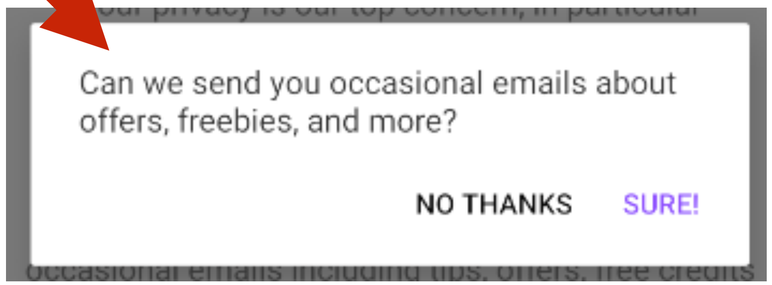
Введение

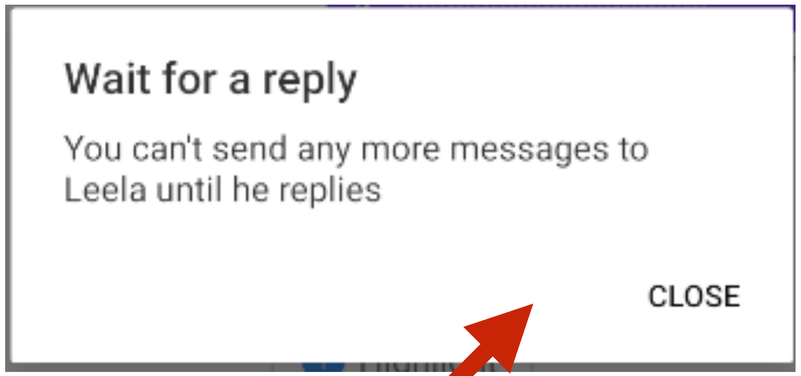

Clear account
Are you sure you want to clear your messages, profile visits, likes and favourites permanently?
CANCEL
OK
Delete
Are you sure you want to delete your only profile photo?
CANCEL
DELETE
Can we send you occasional emails about offers, freebies, and more?
NO THANKS
SURE!
Wait for a reply
You can't send any more messages to Leela until he replies
CLOSE
No internet connection
Unable to reach Badoo server. Please try again later.
CLOSE
Are you sure you’d like to update your gender identity? You can only do this once – if you change it now, you’ll also lose all your matches
GO BACK
I UNDERSTAND

Введение

- Что такое “необязательные” элементы?







Задача

- Создать общий метод верификации всех страниц, основанных на базовой

Пример

```
class ClearAccountAlert < AppAlertAndroid
  def verify_alert_elements
    verify_alert(title:         ClearAccount::TITLE,
                 description:   ClearAccount::MESSAGE,
                 first_button:  ClearAccount::OK_BUTTON,
                 last_button:   ClearAccount::CANCEL_BUTTON)
  end
```

Пример

```
class ClearAccountAlert < AppAlertAndroid
  def verify_alert_lexemes
    verify_alert(title:         ClearAccount::TITLE,
                 description:   ClearAccount::MESSAGE,
                 first_button:  ClearAccount::OK_BUTTON,
                 last_button:   ClearAccount::CANCEL_BUTTON)
  end


class WaitForReplyAlert < AppAlertAndroid
  def verify_alert_lexemes
    verify_alert(title:         WaitForReply::TITLE,
                 description:   WaitForReply::MESSAGE,
                 first_button:  WaitForReply::CLOSE_BUTTON)
  end
end
```

Пример

```
class ClearAccountAlert < AppAlertAndroid
  def verify_alert_lexemes
    verify_alert(title:        ClearAccount::TITLE,
                 description:  ClearAccount::MESSAGE,
                 first_button: ClearAccount::OK_BUTTON,
                 last_button:  ClearAccount::CANCEL_BUTTON)
  end


class WaitForReplyAlert < AppAlertAndroid
  def verify_alert_lexemes
    verify_alert(title:        WaitForReply::TITLE,
                 description:  WaitForReply::MESSAGE,
                 first_button: WaitForReply::CLOSE_BUTTON)
  end
end


class SpecialOffersAlert < AppAlertAndroid
  def verify_alert_lexemes
    verify_alert(description:  SpecialOffers::MESSAGE,
                 first_button: SpecialOffers::SURE_BUTTON,
                 last_button:  SpecialOffers::NO_THANKS_BUTTON)
  end
end
```

Пример

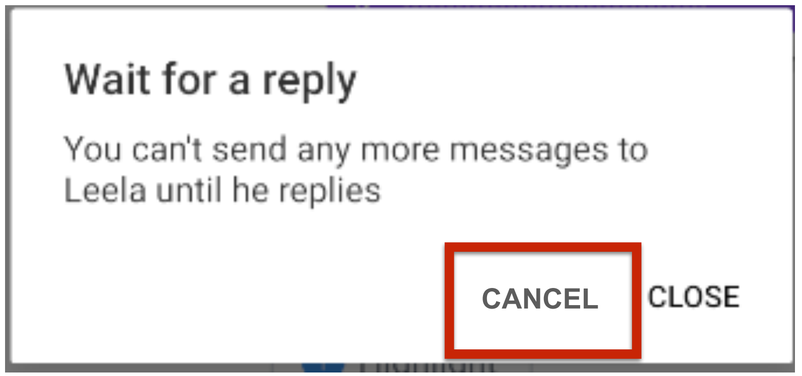
```
def verify_alert(title: nil, description:, first_button:, last_button: nil)
  ui.wait_for_elements_displayed([MESSAGE, FIRST_ALERT_BUTTON])

  ui.wait_for_element_text(expected_lexeme: title, locator: ALERT_TITLE) if title
  ui.wait_for_element_text(expected_lexeme: description, locator: MESSAGE)
  ui.wait_for_element_text(expected_lexeme: first_button, locator: FIRST_ALERT_BUTTON)
  ui.wait_for_element_text(expected_lexeme: last_button, locator: LAST_ALERT_BUTTON) if last_button
end
```

Пример

```
def verify_alert(title: nil, description:, first_button:, last_button: nil)
  ui.wait_for_elements_displayed([MESSAGE, FIRST_ALERT_BUTTON])

  ui.wait_for_element_text(expected_lexeme: title, locator: ALERT_TITLE) if title
  ui.wait_for_element_text(expected_lexeme: description, locator: MESSAGE)
  ui.wait_for_element_text(expected_lexeme: first_button, locator: FIRST_ALERT_BUTTON)
  ui.wait_for_element_text(expected_lexeme: last_button, locator: LAST_ALERT_BUTTON) if last_button
end
```

Проблема

- **Неполная проверка необязательных элементов**



Пример решения

```
ui.wait_for_element_text(expected_lexeme: title, locator: ALERT_TITLE) if title
```

```
if title.nil?
  Assertions.assert_false(ui.elements_displayed?(ALERT_TITLE),
                          “Alert title should not be displayed")
else
  ui.wait_for_element_text(expected_lexeme: title, locator: ALERT_TITLE)
end
```

Пример решения

```ruby
def wait_for_optional_element_text(expected_lexeme:, locator:)
  GuardChecks.not_nil(locator, 'Locator should be specified')

  if expected_lexeme.nil?
    Assertions.assert_false(elements_displayed?(locator), "Element with locator #{locator} should not be displayed")
  else
    wait_for_element_text(expected_lexeme: expected_lexeme, locator: locator)
  end
end
```

Пример решения

```
def verify_alert(title: nil, description:, first_button:, last_button: nil)
  ui.wait_for_elements_displayed([MESSAGE, FIRST_ALERT_BUTTON])

  ui.wait_for_optional_element_text(expected_lexeme: title, locator: ALERT_TITLE)
  ui.wait_for_element_text(expected_lexeme: description, locator: MESSAGE)
  ui.wait_for_element_text(expected_lexeme: first_button, locator: FIRST_ALERT_BUTTON)
  ui.wait_for_optional_element_text(expected_lexeme: last_button, locator: LAST_ALERT_BUTTON)
end
```

Вывод

- Необходимо проверять все состояния “необязательных” элементов
- Полезно создать общий метод для всех страниц

Применение

- Используйте полную систему проверок
- Выделяйте общий метод для однотипных действий

Итоги

Итоги

- Используйте полную систему проверок
- Добавляйте проверку предустановки для асинхронных действий
- Выделяйте общие методы для переиспользования однотипного кода
- Создавайте объект тестирования простым
- Выделяйте независимые методы для простых действий в тестах

СПАСИБО!

dmitrii.makarenko@team.bumble.com
v.karanevich@team.bumble.com
tech.badoo.com

Ссылки

https://github.com/badoo/MobileAutomationSampleProject