Drive a common sign-up flow across native Web, iOS and Android

Máté Marjai - SumUp

About us

SumUp is a rapidly expanding mobile payments startup. Our aim is to help small businesses to accept card payments through their mobile devices without the need of costly bank and merchant setup.

We are changing the way people transact to a frictionless experience for both parties - merchant and customer.


Account
Coffee & Cream
Clear
Cappuccino
4,00 £
Bagel
2,95 £
Panini
4,20 £
Total
11,15 £
Sales tax incl.
Charge
Drinks
Manual entry
Cappuccino
Espresso
Juice
Macchiato
Milk Shake
Water
A-Z
Drinks
Food
Z

A short history

- SumUp launched in 4 countries in August 2012: Ireland, UK, Germany, Austria
- Expanded to 3 more in October: Italy, Netherlands, Spain
- Expanded to an additional 3 in December: France, Belgium, Portugal

The pain - localization

With the rapid expansion we had to tailor our sign-up flow for every country. I.e.:

- Bank fields are different per country
- Tax/VAT registration number regexes
- Address fields
- Name fields

The pain - change risk

These little differences meant changes in 5 components:

- iOS app
- Android app
- Web
- Core API
- Database

The pain - change risk

Changing 5 things - a lot can go wrong, and we all heard about Murphy's law:

- Error messages are not matching up
- Regexes are not matching up

The pain - time

- Dev time on apps and web essentially doing the same thing.
- QA time on all platforms.
- Align releases around the same time for all components
- More components - more time spent on bug fixing.

The pain - analytics

Flying half-blind: we had to make assumptions about where the dropping points were during the flow for our merchant-to-bes.

The original implementation wasn't flexible enough to do real improvements and this was affecting our conversion rates.

The pain - releases

Release and review times are precious. With Apple potentially sitting on an app review for week(s) is not ideal.

Releasing 5 components - higher chance of breaking something by missing a merge, release etc...


We should fix our sign-ups
quickmeme.com

The spark - the idea

Started brainstorming with app devs about how we could send a list of fields and regexes to the apps to collect during the flow.

After some discussions we realised - why not go the extra mile and describe the screens/pages themselves?

The spark - timing

Discussions in October, but we had tight deadlines. In a startup it's always a battle between dev cost ('build it right') and delivery time ('build it fast'). Idea was parked for later.

The securing of our full payments institution licence from the UK's Financial Services Authority in Jan 2013 led us to revamp our sign-up flow. So we thought: this is our opportunity!


ONE FLOW
FURR ALL

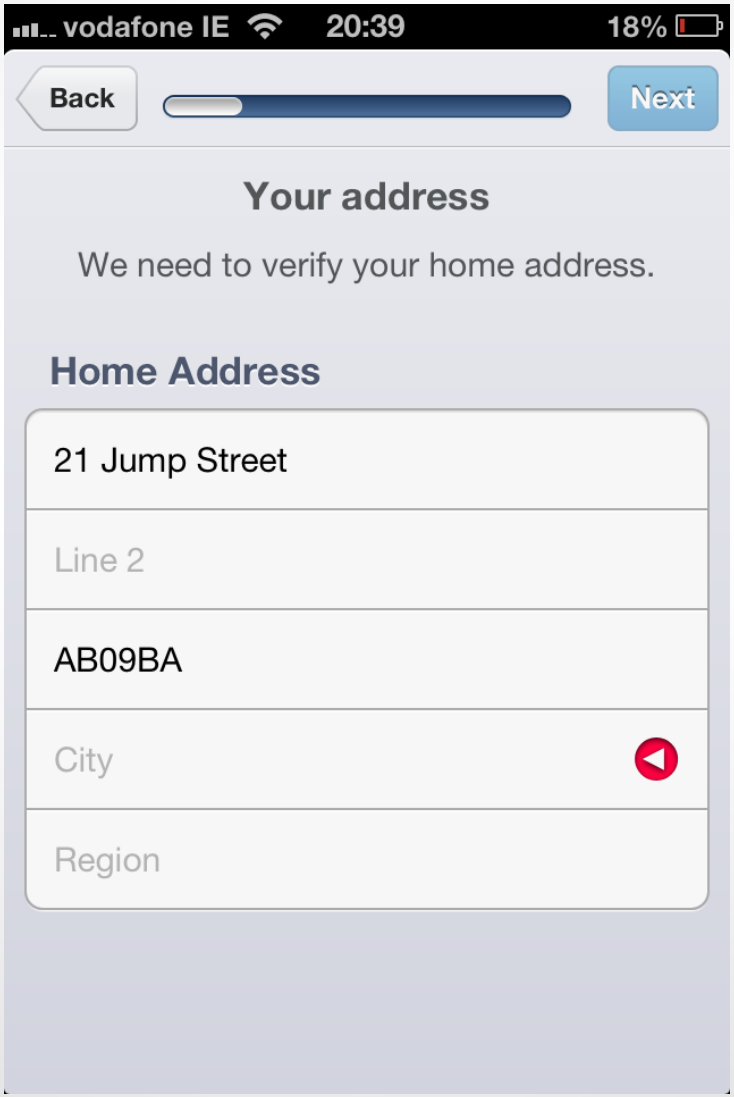
The requirements

- Centralise sign-ups from our Core API to all platforms
- Create an easily mouldable solution for the forms (switch steps around, etc...)
- Describe our sign-up steps that are generic enough for 3 platforms: Web, iOS, Android
- Better tracking for sign-ups
- It needs to be extensible for the next N launch countries

Motifs

- Help the business with quicker response time on flow changes.
- To have a flow that behaves exactly the same way on all platforms
- Have an easy way to describe steps with a language that all devices can interpret and turn into a native form.
- Less Dev and QA time spent on minor details like a regex change.

How would you describe a page?



Create a free SumUp Account

Your details

This information is required to verify your identity.

Business

Full name of the business

Website of the business

Personal Details

First name

Last name

Date of Birth

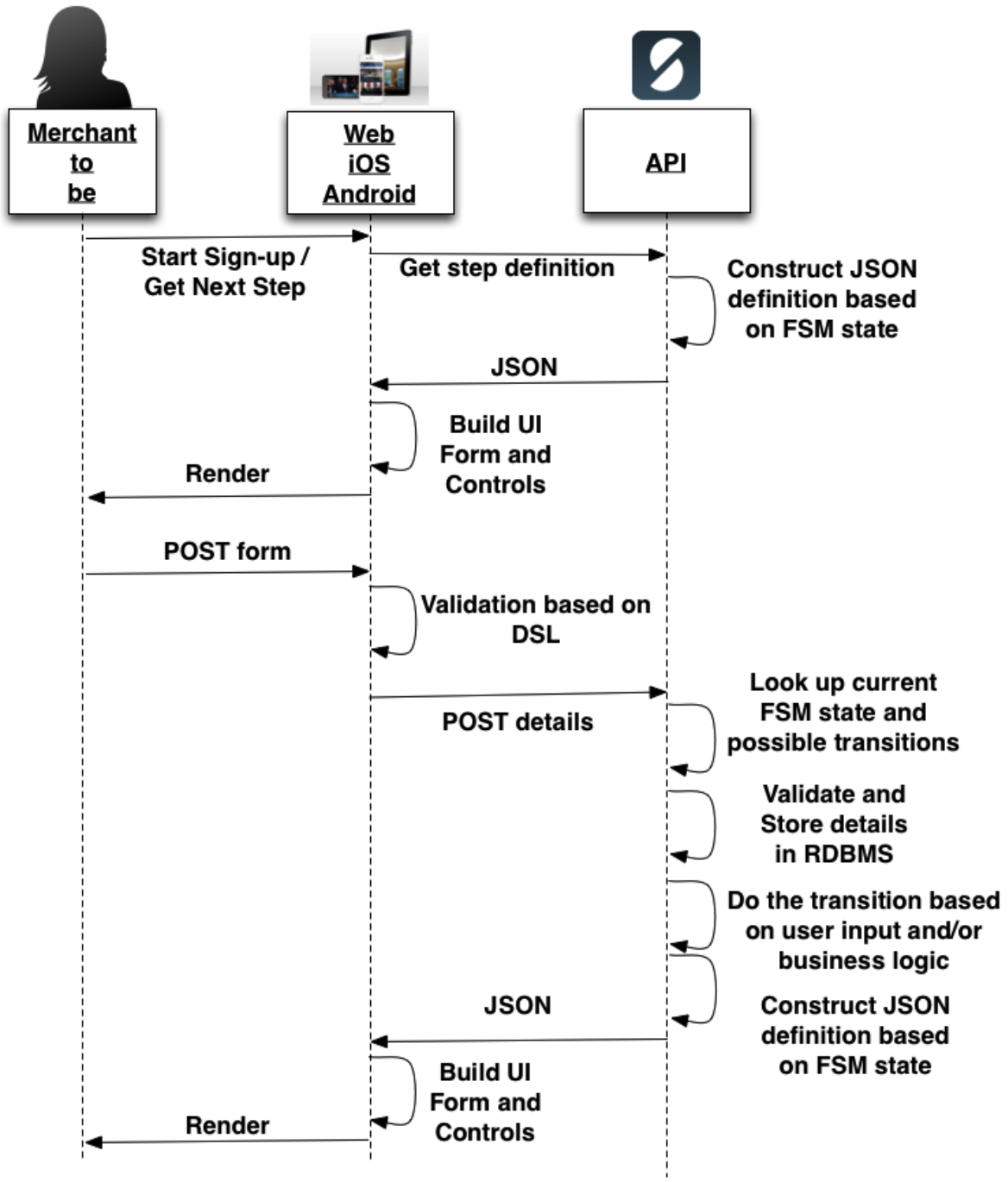
Mobile phone number

Home phone number (optional)

Next

The flow

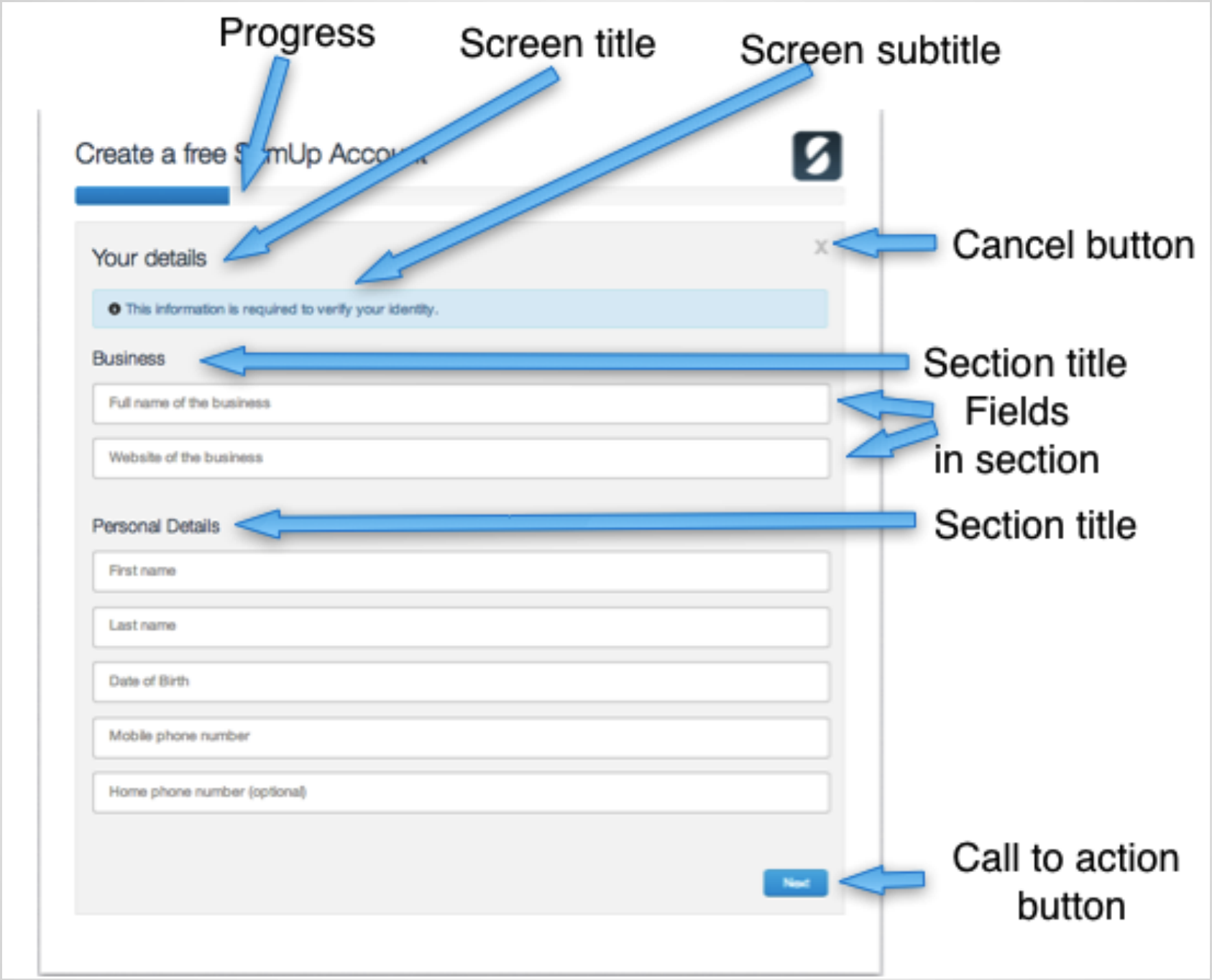
- Sign-up flow can be portrayed with a finite state machine.
- Each state has a corresponding step to complete to make the transition(s).
- Multiple options on transitions - skippable steps, different actions, background ID checks, etc...


Merchant to be
Web iOS Android
API
Start Sign-up / Get Next Step
Get step definition
Construct JSON definition based on FSM state
JSON
Build UI Form and Controls
Render
POST form
Validation based on DSL
POST details
Look up current FSM state and possible transitions
Validate and Store details in RDBMS
Do the transition based on user input and/or business logic
Construct JSON definition based on FSM state
JSON
Build UI Form and Controls
Render

DSL - high level

Each flow Step can have the following:

- List of Screens
  - Screen Metadata
  - List of Sections
    - Section Metadata
    - List of Fields
      - Field metadata
      - Field validations
  - List of Buttons
- Step buttons
  - Skip button
  - Cancel button
  - Progress


Progress
Screen title
Screen subtitle
Create a free SumUp Account
Your details
x
Cancel button
This information is required to verify your identity.
Business
Section title
Full name of the business
Website of the business
Fields
in section
Personal Details
Section title
First name
Last name
Date of Birth
Mobile phone number
Home phone number (optional)
Next
Call to action
button

DSL - Screen

```
"screens": [
    {
      "ref": "screen_id",
      "progress": 10,
      "type": "generic_table",
      "title": "Screen title",
      "subtitle": "Subtitle",
      "sections": [ ... ],
      "buttons": [ ... ]
    }
]
```

DSL - Section

```
"sections": [
  {
    "ref": "section_id",
    "optional": true,
    "hidden": false,
    "title": "Section title",
    "fields": [ ... ],
    "url": "https://sumup.com/faq",
    ...
```

DSL - Section

```
    ...
    "toggle_controls": {
      "default_state": false,
      "true_text": "Yes",
      "false_text": "No",
      "message": "Toggle"
    },
    "toggle_conditions":[
      {
        "field_ref": "business_type",
        "values": ["Other Retail"]
      },
    ],
  },
]
```

DSL - Field

```
{
   "ref": "name",
   "title": "Field A",
   "type": "text",
   "keyboard_type": "default",
   "disabled": "false",
   "default_value": "John Smith",
   "possible_values": [ ... ],
   "placeholder_text":"Enter A",
   "info_url": "https://sumup.com/faq",
   "validation": { ... }
}
```

DSL - Validation

```
"validation": {
    "optional": "false",
    "regex": "/^a[a-zA-Z]+$/",
    "regex_during_typing": "/^a.*$/",
    "min": null,
    "max": null,
    "matches": null
 }
```

DSL - Button

```
{
  "ref": "submit",
  "name": "Submit",
  "primary": "true",
  "prominent": "true",
  "action": "submit",
  "confirmation": { ... }
}
```

DSL - Button notification

```
{
   ...
   "confirmation": {
      "show": "false",
      "notification": {
         "title": "Confirm submit",
         "message": "Please make sure
all details submitted are valid..."
      }
   }
}
```

DSL - Error messages

```
{
  "code": 1234,
  "data": {
    "field_validations": [
      {
        "field_ref": "first_name",
        "notification": {
          "message": "UH-OH!",
          "title": "First Name?"
        }
      },
    ],
  },
  "message": "Failed validation message!"
}
```

DSL - Special cases

- Cancel and skip buttons
- Some less generic screens - ie Business owners screen.
- Keyboard types

Impacts

- Completely new sign-up flows in just 3 days.
- Less coding, less dev, less testing per new flow.
- Finite sign-up states = better tracking of our merchants and their behaviour during the flow.
- Improved conversion rates for sole traders. (more than doubled conversion in the UK!)

What's next

- Further improvements on the JSON generation.
- Respect formatting differences between native interfaces (iOS, Android, Web) - interface-aware screen definitions.
- Drive our checkout flow with this DSL.
- Clean up and open source - We think there might be others who could use this solution.

Kudos

The whole SumUp Software team that helped this to bring live.

BOW