

Les machines à état

une documentation exécutable ?
et debuggable ? et testable ?



Antoine Cailly



acailly



@AntoineCailly



antoine.cailly@zenika.com

Pourquoi ce talk ?



Cahier des charges
= *ce que fait l'application*



Cahier des charges
= *ce que fait l'application*



Spécifications fonctionnelles
= *ce que fait l'application*



Cahier des charges
= ce que fait l'application



Spécifications fonctionnelles
= ce que fait l'application



Cahier de recette
= ce que fait l'application

A la recherche de...



Fonctionnel



Dev



???

= ce que fait l'application



Test



Doc



Fonctionnel



Dev



Machine à état ?
= ce que fait l'application



Test



Doc

- Les machines à état c'est quoi ?
- Est ce que c'est adapté à mon contexte ?

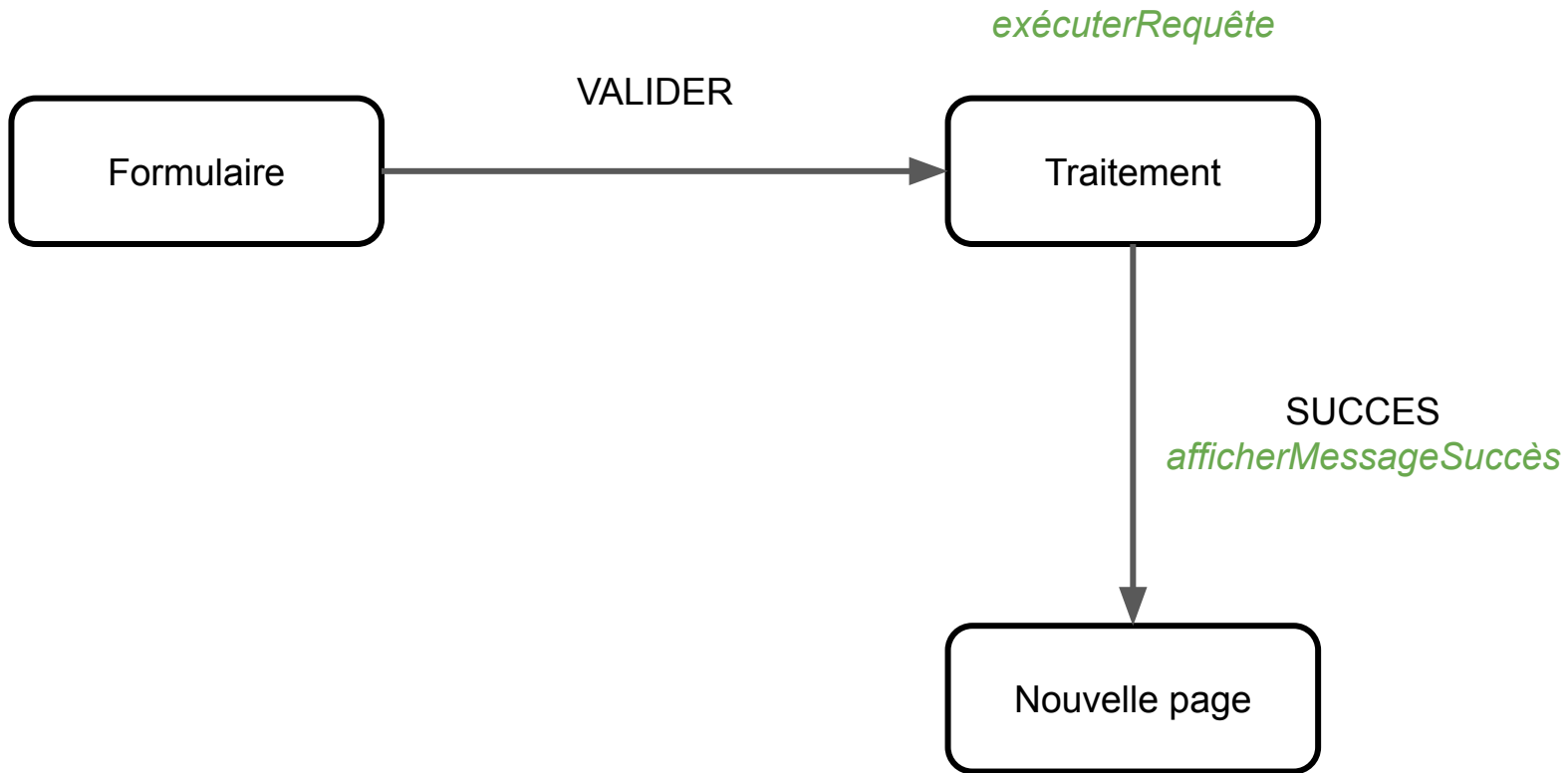
Les machines à état c'est quoi ?

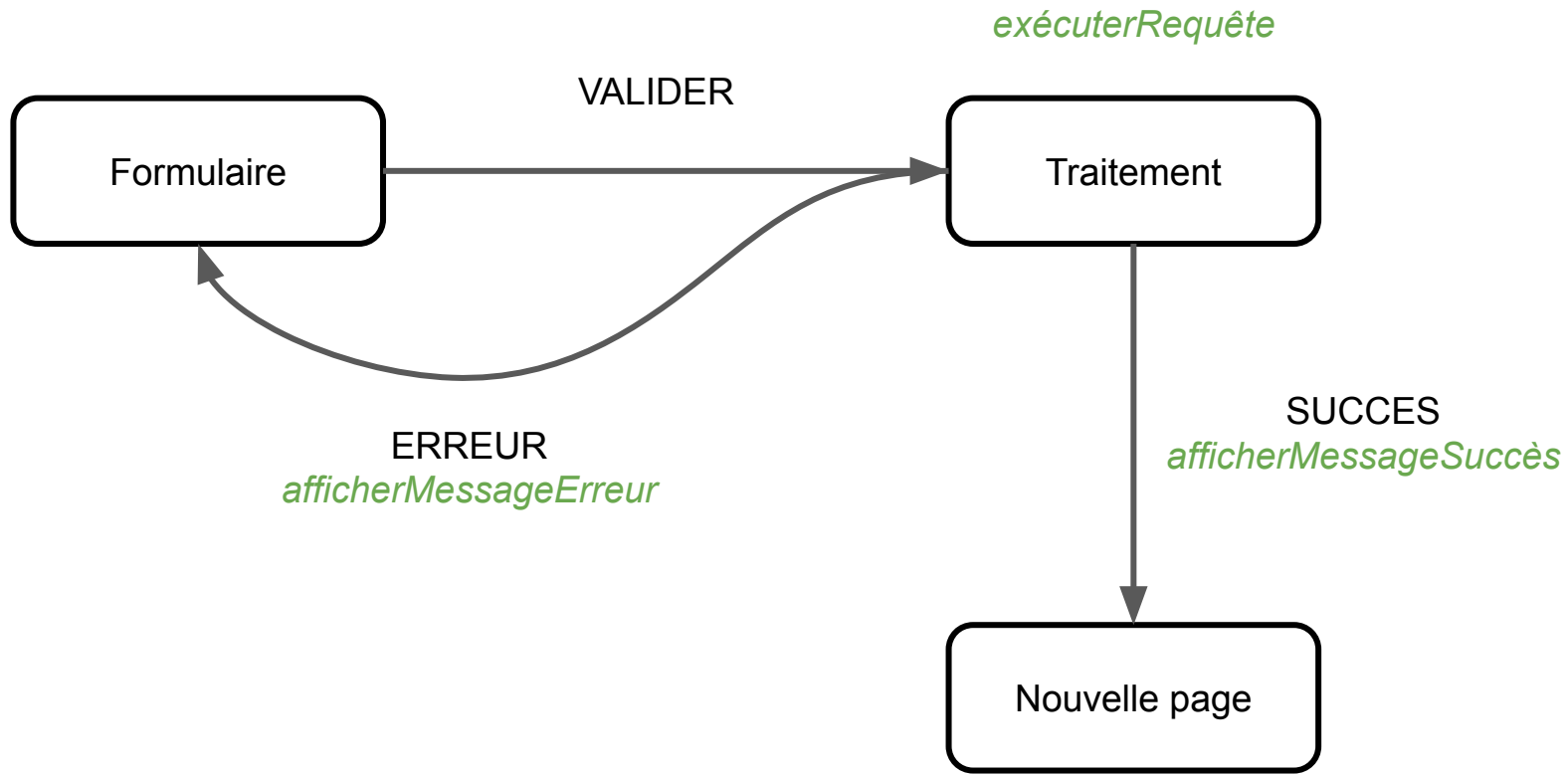
Démo Stately

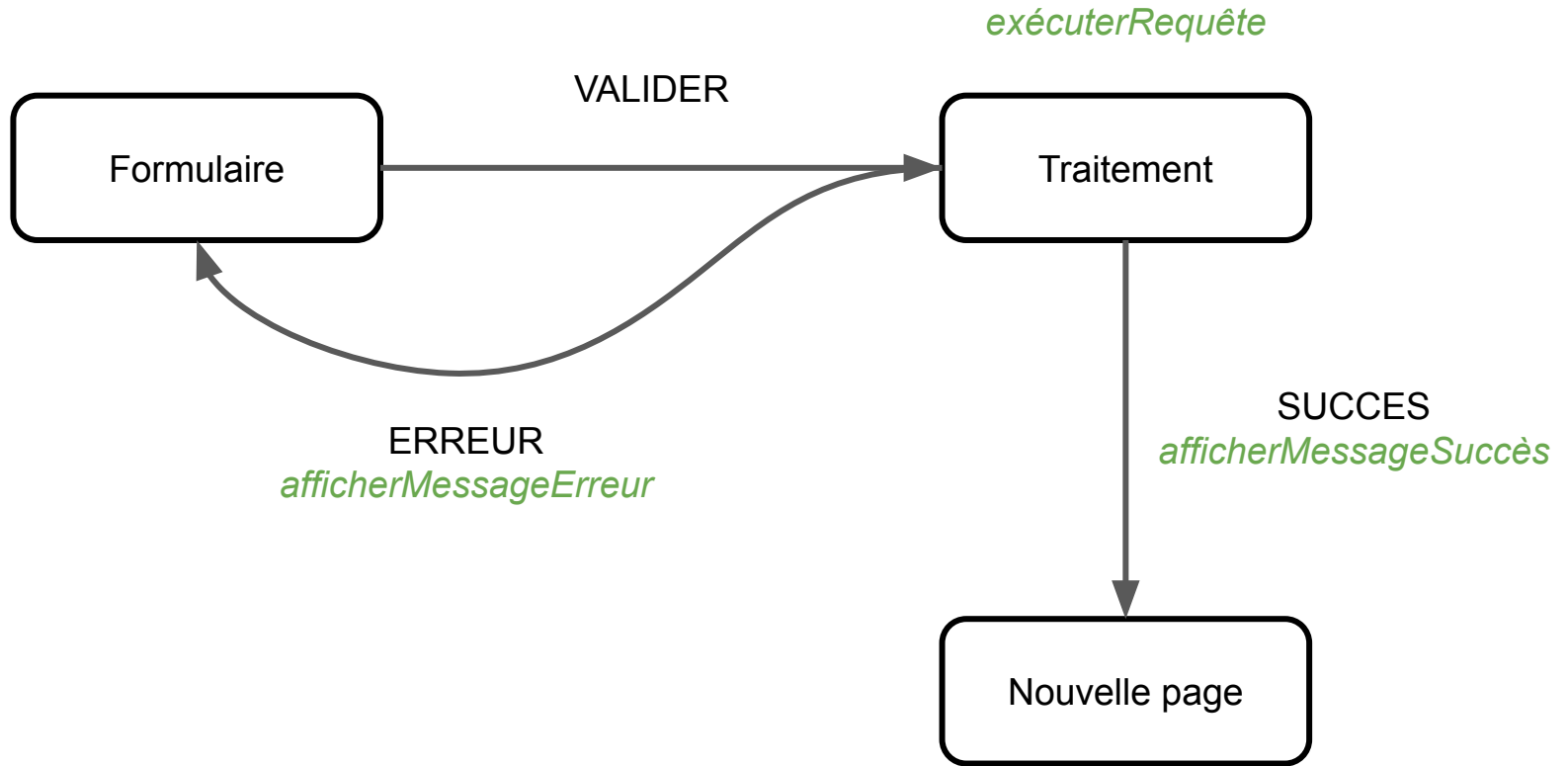
<https://state.new/>

Formulaire









State(S) x Event(E) -> State(S'), Actions (A)

Les machines à état sont partout !



Fonctionnel 



Dev



Machine à état ?
= *ce que fait l'application*



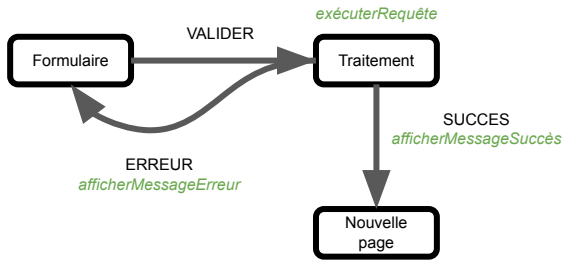
Test



Doc

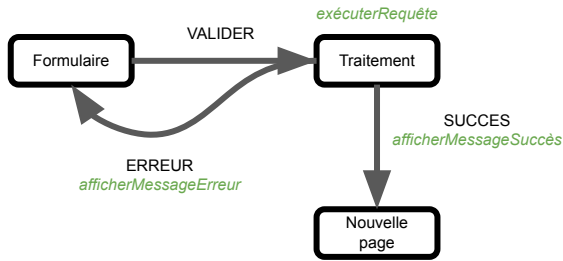
Ensuite, les devs





```

{
  "id": "D mo Breizhcamp",
  "initial": "Formulaire",
  "states": {
    "Formulaire": {
      "on": {
        "Valider": {
          "target": "Traitement"
        }
      }
    },
    "Traitement": {
      "invoke": {
        "src": "ex cuterRequ te",
        "onDone": [
          {
            "target": "Nouvelle page",
            "actions": "afficherMessageSucc s"
          }
        ],
        "onError": [
          {
            "target": "Formulaire",
            "actions": "afficherMessageErreur"
          }
        ]
      }
    },
    "Nouvelle page": {}
  }
}
  
```

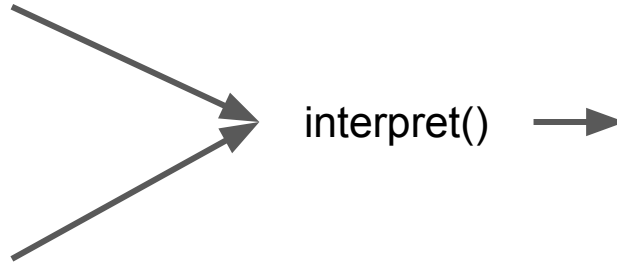


Machine à état

+

Actions déclenchées
par la machine à état

exécuterRequête = ...
afficherMessageSuccès = ...
afficherMessageErreur = ...



interpret()

Machine à état
exécutable

Démo Inspector



Fonctionnel 



Dev 



Machine à état ?
= ce que fait l'application



Test



Doc

Et maintenant, le test

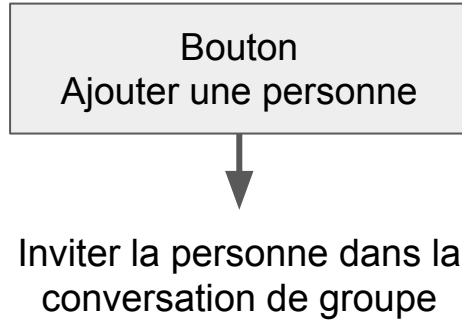
Laissez moi vous conter une histoire...

*En tant qu'utilisateur,
quand je swipe up je peux cliquer sur le bouton "Ajouter une personne"
afin d'ajouter une personne dans la conversation de groupe*

Bouton
Ajouter une personne

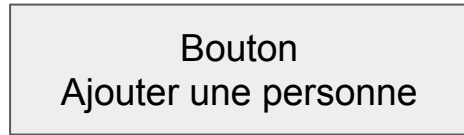
*En tant qu'utilisateur,
quand je swipe up je peux cliquer sur le bouton "Ajouter une personne"
afin d'ajouter une personne dans la conversation de groupe*

Bouton
Ajouter une personne



Inviter la personne dans la
conversation de groupe

*En tant qu'utilisateur,
quand je swipe up je peux cliquer sur le bouton "Ajouter une personne"
afin d'ajouter une personne dans la conversation de groupe*



Inviter la personne dans la
conversation de groupe

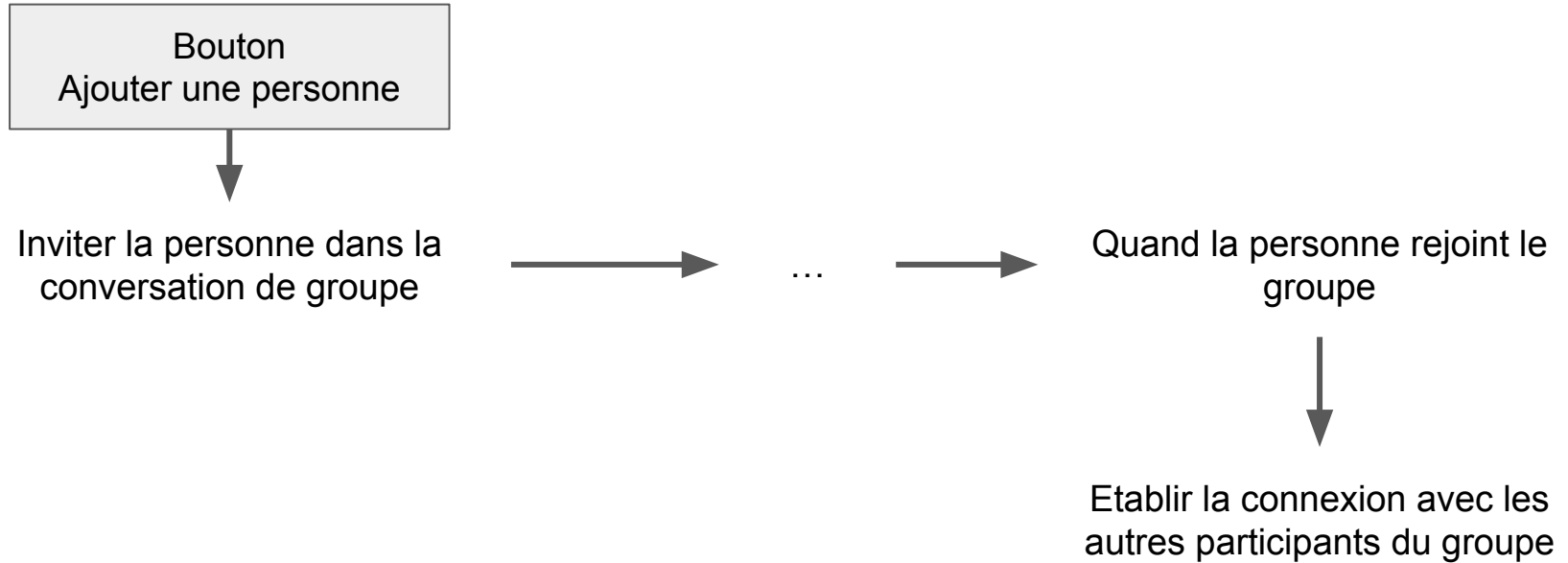


...



Quand la personne rejoint le
groupe

*En tant qu'utilisateur,
quand je swipe up je peux cliquer sur le bouton "Ajouter une personne"
afin d'ajouter une personne dans la conversation de groupe*

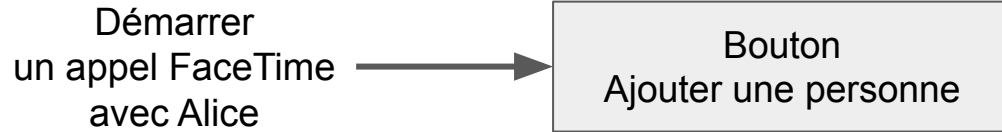


quand tout à coup !

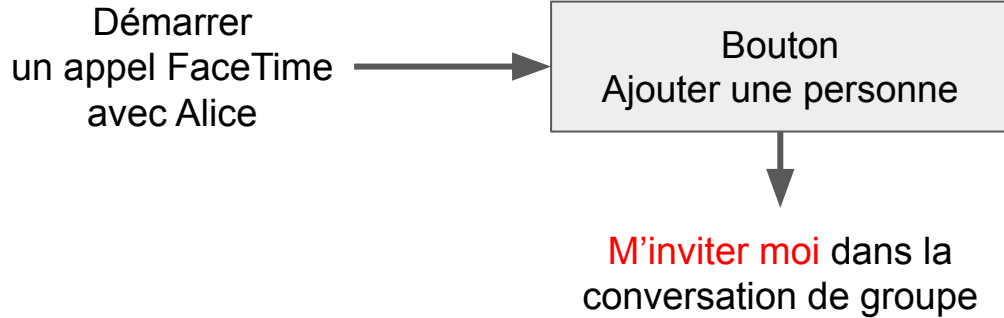
*En tant qu'utilisateur,
quand je swipe up je peux cliquer sur le bouton "Ajouter une personne"
afin d'ajouter une personne dans la conversation de groupe*

Démarrer
un appel FaceTime
avec Alice

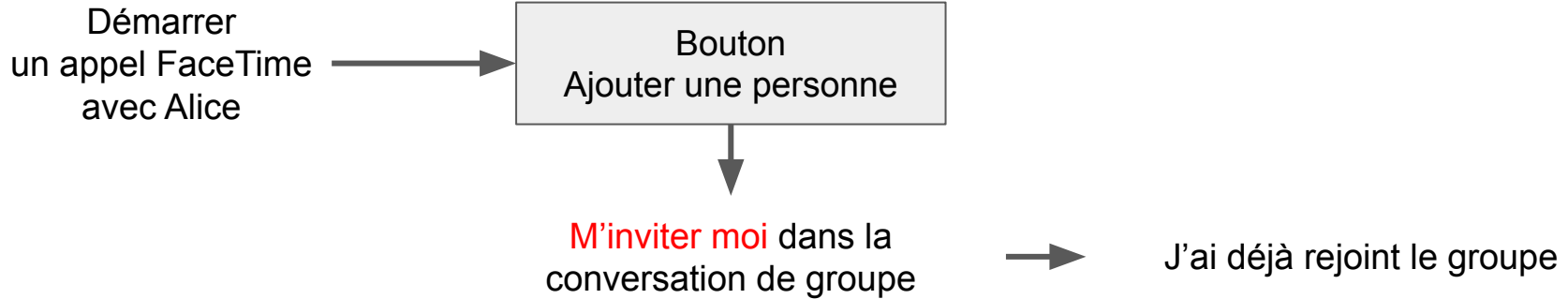
*En tant qu'utilisateur,
quand je swipe up je peux cliquer sur le bouton "Ajouter une personne"
afin d'ajouter une personne dans la conversation de groupe*



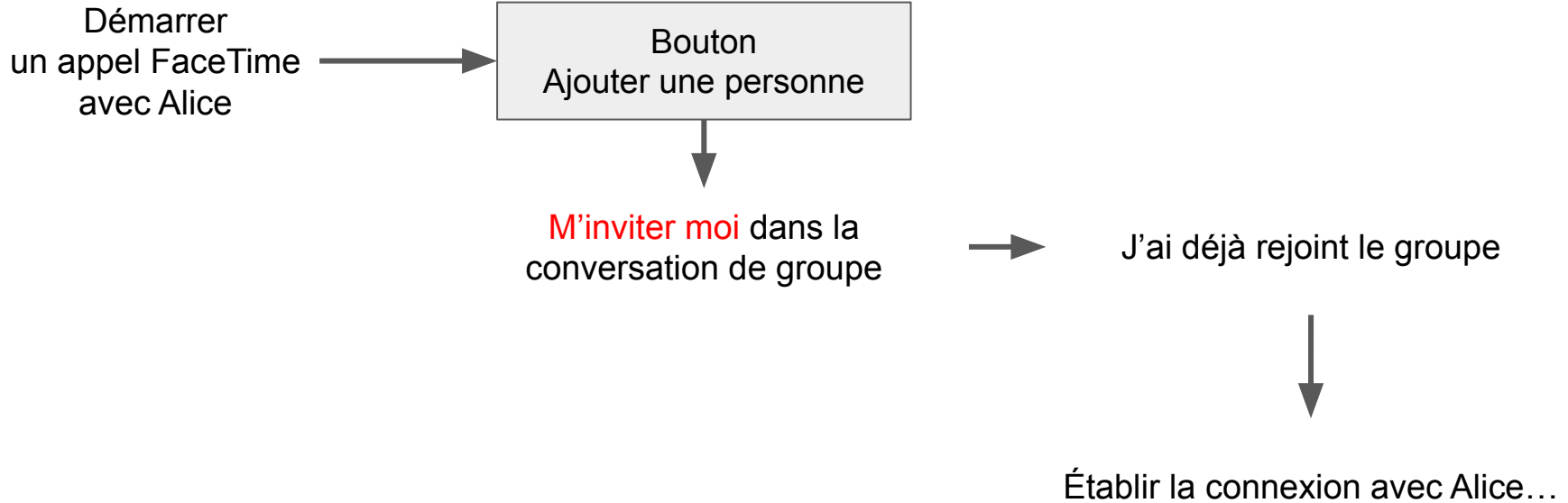
*En tant qu'utilisateur,
quand je swipe up je peux cliquer sur le bouton "Ajouter une personne"
afin d'ajouter une personne dans la conversation de groupe*



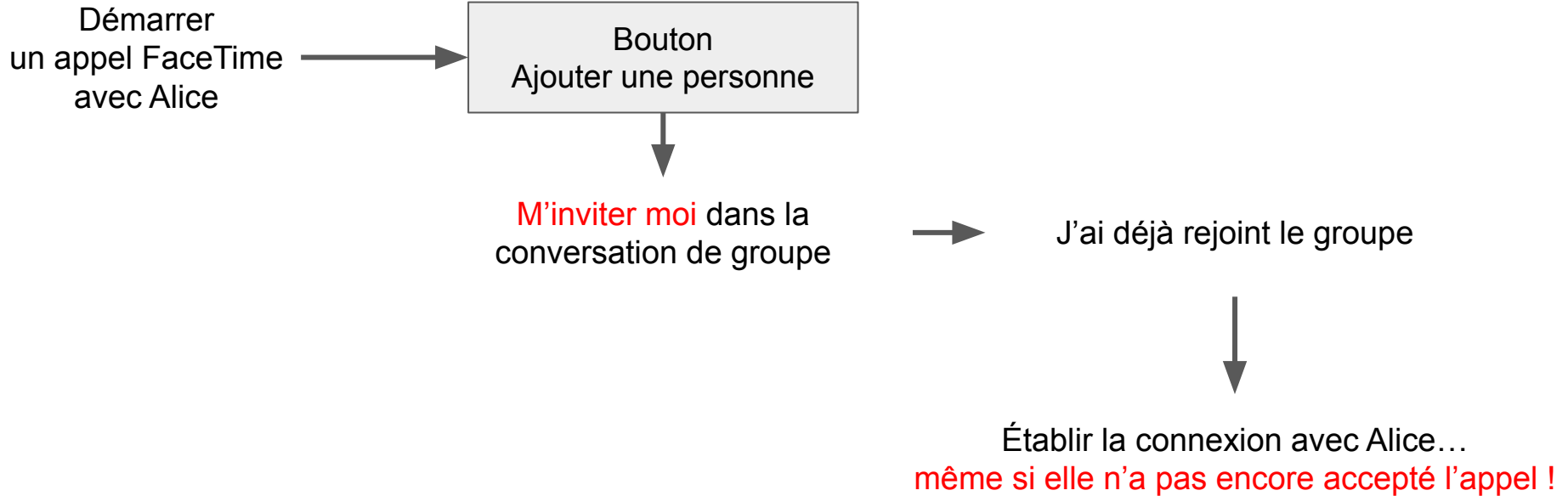
*En tant qu'utilisateur,
quand je swipe up je peux cliquer sur le bouton "Ajouter une personne"
afin d'ajouter une personne dans la conversation de groupe*



*En tant qu'utilisateur,
quand je swipe up je peux cliquer sur le bouton "Ajouter une personne"
afin d'ajouter une personne dans la conversation de groupe*



*En tant qu'utilisateur,
quand je swipe up je peux cliquer sur le bouton "Ajouter une personne"
afin d'ajouter une personne dans la conversation de groupe*



State explosion



Personne appelant

Off

Ca sonne

Contact établi

X



Personne appelée

Off

Ca sonne

Contact établi

X



Personne ajoutée

Off

Ca sonne

Contact établi

State explosion



Personne appelant

Off

Ca sonne

Contact établi

X



Personne appelée

Off

Ca sonne

Contact établi

X



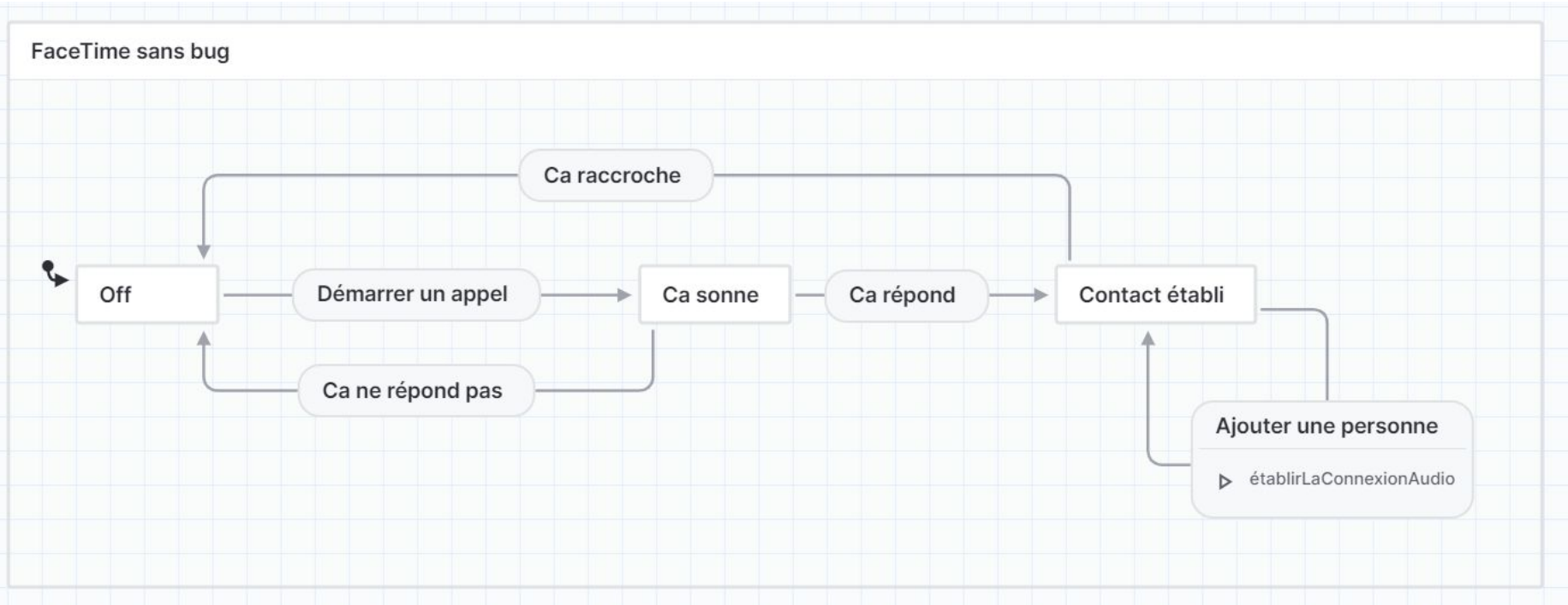
Personne ajoutée

Off

Ca sonne

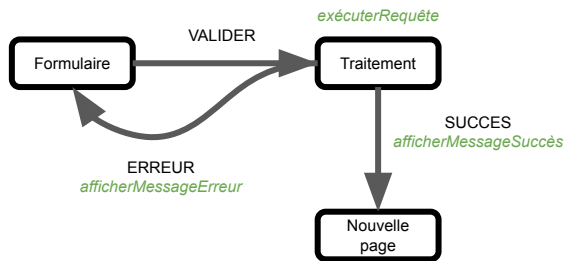
Contact établi

Implicit vs explicit state



La machine a état déclenche l'ajout d'une personne **uniquement** depuis l'état contact établi

Ca ne sert à rien de tester un chemin qui
n'est pas dans la machine 🤪



Machine à état

Événement



Etat

Interface utilisateur

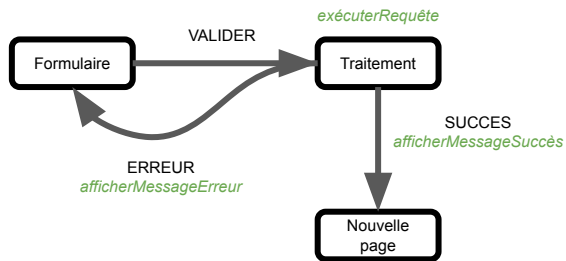
Interagit



Vérifie

Testeur.euse

On peut automatiser
les interactions et la vérification
avec les tests End-To-End (E2E) 🌟😄



Machine à état

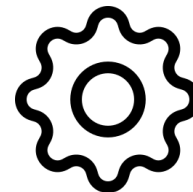
Événement



Etat

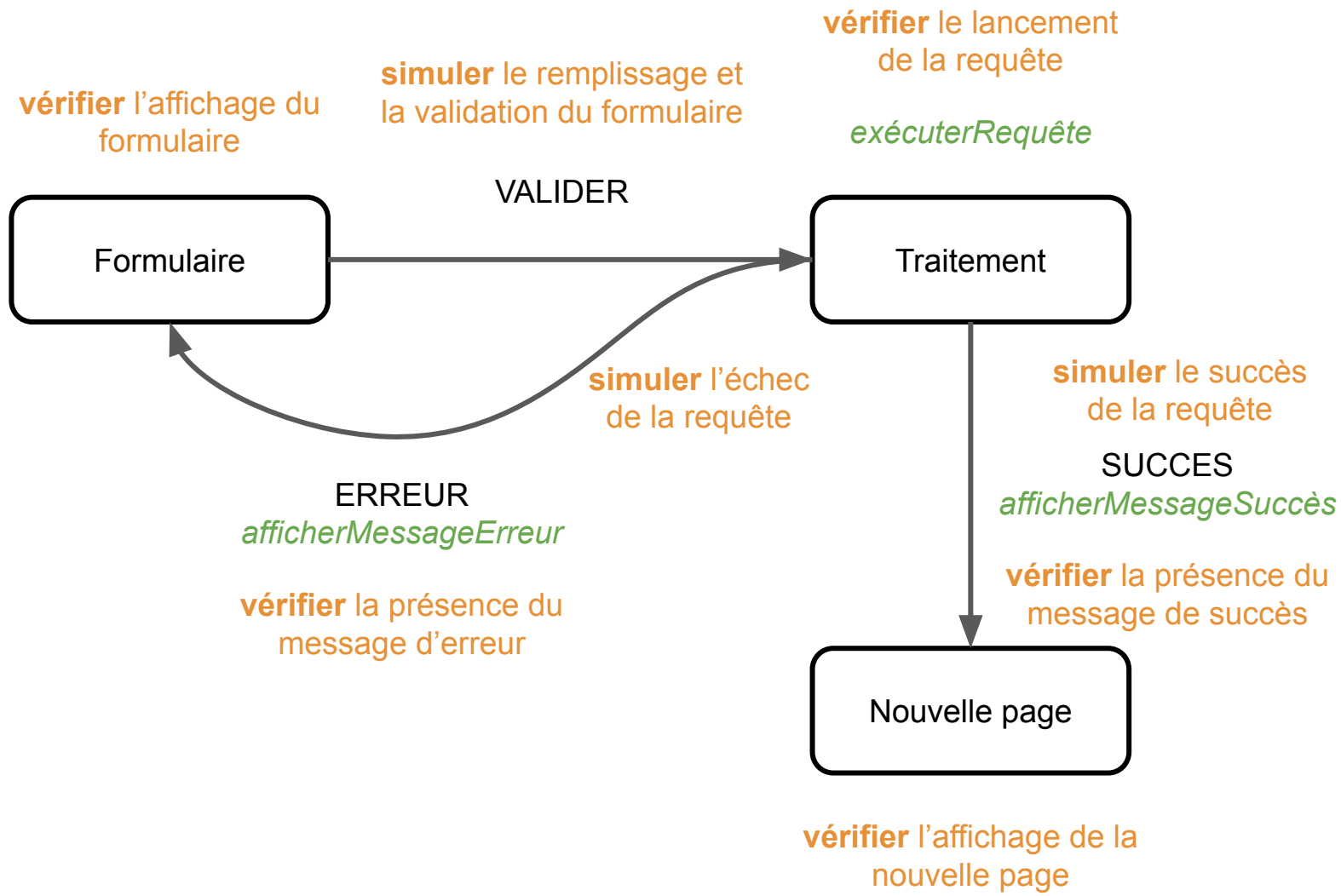
Interface utilisateur

Interagit



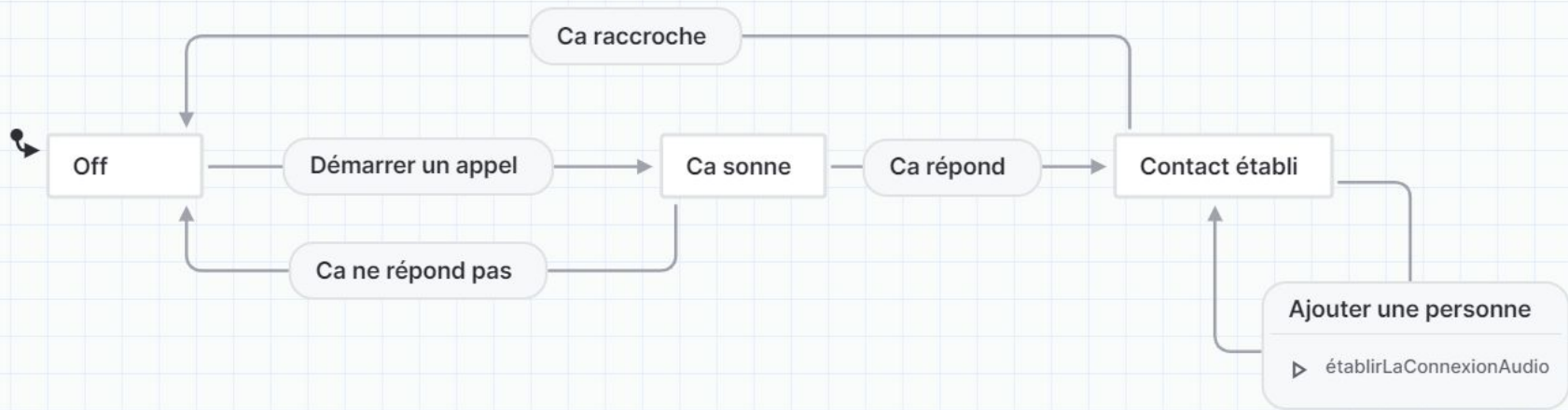
Vérifie

Cypress



On peut automatiser
le parcours de la machine à état 🤖

FaceTime sans bug



Passer par tous les états au moins une fois, passer par tous les chemins au moins une fois, etc.

Démo Model Based Testing

```
const testMachine = addTests(machine, testStates)
const testModel = createModel(testMachine).withEvents(testEvents)

describe('abonnements', () => {
  beforeEach(function () {
    cy.visit('http://localhost:3000/feeds')
    cy.contains('Charger les données fictives').click()
  })

  const testPlans = testModel.getSimplePathPlans()
  testPlans.forEach((plan) => {
    describe(plan.description, () => {
      plan.paths.forEach((path) => {
        it(path.description, () => {
          cy.then(path.test)
        })
      })
    })
  })
})

describe('coverage', () => {
  it('should pass', () => {
    testModel.testCoverage()
  })
})
})
```





Fonctionnel 



Dev 



Machine à état ?
= ce que fait l'application



Test 



Doc

Enfin, la doc

Démo génération Mermaid et Markdown

<https://github.com/acailly/state-machine-rss-reader/blob/main/src/abonnements/abonnements.machine.doc.test.md>



Fonctionnel 



Dev 



Machine à état ?
= ce que fait l'application



Test 



Doc 




Fonctionnel 



Dev 



Machine à état 
= ce que fait l'application



Test 



Doc 

Est ce que c'est adapté à mon contexte ?

Est ce que c'est adapté à mon contexte ?

*Disclaimer : gardez un esprit critique,
mon avis n'est que mon avis et vous connaissez mieux votre contexte que moi 😊*

REX n° 1 : le mien

Quoi : Un lecteur RSS pour usage perso

Objectif : Est-ce viable pour une application interne avec beaucoup d'intervenants ponctuels avec des compétences et des appétences variées ?

Positif 👍 :

- facilite la compréhension de l'existant
- moins de choses à jeter en cas de refonte

Négatif 👎 :

- Glue entre le JSON et le code parfois fastidieuse à coder
- L'éditeur de statelike ne permet pas d'ajouter/d'afficher tout ce que sait faire xstate

REX n° 2 : un ex-collègue

Quoi : Une application musicale gérant des média et interagissant avec des briques externes

Objectif : Centraliser les interactions en une seule source de vérité visuelle et debuggable

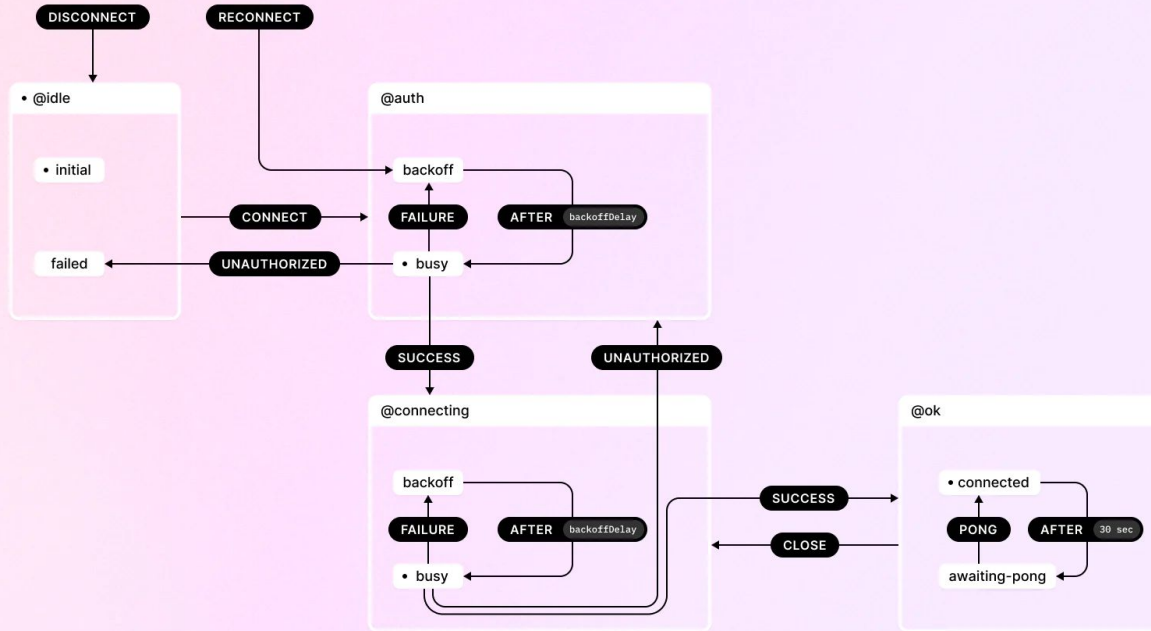
Positif 👍 :

- “Tellement pratique et maintenable alors que l'application se complexifiais”
- Discuter en modifiant la machine sans coder tout ce qu’il y a derrière

Négatif 👎 :

- Nécessité de refondre une partie de l’existant
- “Un peu fastidieux et lourd à écrire”

REX n° 3 : publication de LiveBlocks



By completely rewriting our previous connection engine as a formalized state machine, we have taken a holistic approach to resolve all known bugs around connectivity and client recovery in edge cases [...]

The rewritten connection engine, formalized and implemented as a state machine, streamlines the connection and reconnection process, greatly reducing the number of states we handle internally, making it easier to reason about the exact state of the Liveblocks connection at any given moment.

REX TL;DR; Une histoire de compromis

"Je pense que ça se prête vachement bien à des applications web à interaction forte (type éditeur WYSIWYG, appli avec médias, etc.) mais moins à des sites où toute la logique est portée côté backend"

A utiliser là où il y a beaucoup d'**interactions** 🎯
(même dans le backend)

"Un peu fastidieux et lourd à écrire mais tellement pratique et maintenable alors que l'application se complexifie"

Fiabilité et maintenabilité 👍 vs fastidieux et lourd 👎



Usages connus



BackMarket

Des alternatives moins fastidieuses ?



Robot

Fast **1kB** functional library for creating Finite State Machines

xstate

```
{
  "id": "Démo Breizhcamp",
  "initial": "Formulaire",
  "states": {
    "Formulaire": {
      "on": {
        "Valider": {
          "target": "Traitement"
        }
      }
    },
    "Traitement": {
      "invoke": {
        "src": "exécuterRequête",
        "onDone": [
          {
            "target": "Nouvelle page",
            "actions": "afficherMessageSuccès"
          }
        ],
        "onError": [
          {
            "target": "Formulaire",
            "actions": "afficherMessageErreur"
          }
        ]
      }
    },
    "Nouvelle page": {}
  }
}
```

robot

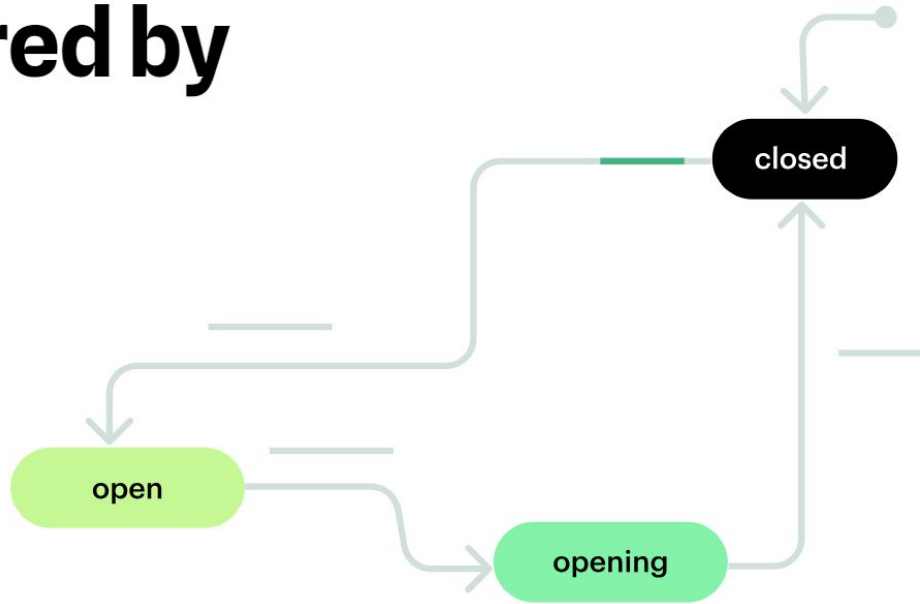
```
createMachine(
  "Formulaire",
  {
    Formulaire: state(
      transition("Valider", "Traitement")
    ),
    Traitement: invoke(
      exécuterRequête,
      transition(
        "done", "Nouvelle page",
        afficherMessageSuccès
      ),
      transition(
        "error", "Formulaire",
        afficherMessageErreur
      )
    ),
    "Nouvelle page": state()
  }
)
```

 *exécuterRequête,
afficherMessageSuccès et
afficherMessageErreur sont des fonctions
(pas de mapping à faire)*

UI components powered by Finite State Machines

A collection of framework-agnostic UI component patterns like **accordion**, **menu**, and **dialog** that can be used to build design systems for React, Vue and Solid.js

Get Started →



La version de la simplification

May 25, 2023 — 14 minute read

Announcing XState v5 beta



David Khourshid

Stately Team

<https://stately.ai/blog/announcing-xstate-v5-beta>

Merci 🙏

<https://github.com/acailly/state-machine-rss-reader>



acailly



@AntoineCailly



antoine.cailly@zenika.com

