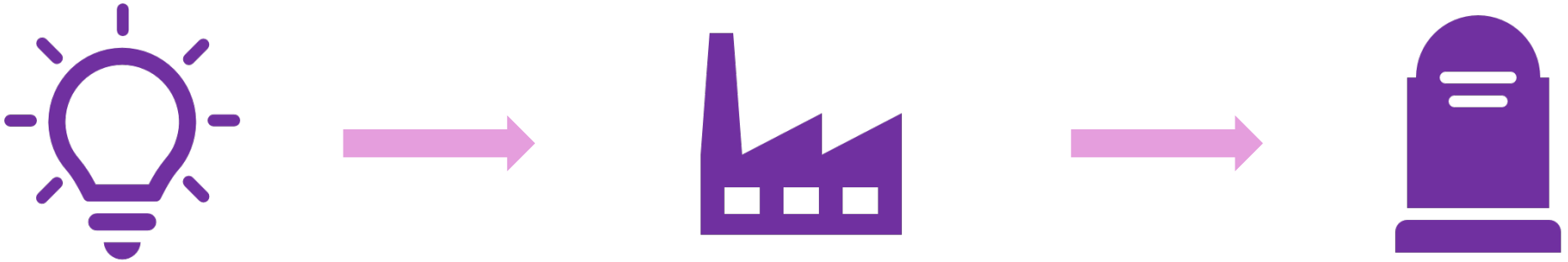
cognizant softvision

Elixir Lang
Now & beyond

Start simple, go far

March 2023, updated June 2024

Product Lifecycle



Start

Idea

Continue

POC

MVP

Release

...

Stop

Decommissioning

cognizant softvision

This is the situation - 1st iteration

👩‍💻 Product Owner

Asks for an POC to be made.

Title: Counter example web app

Specs: Should INC/DEC

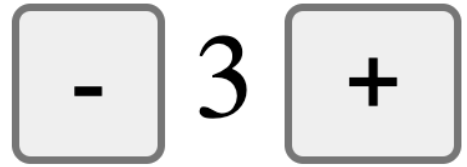
👨‍💻 Developers

Team structure:

- 1 Ruby dev
- 1 React dev
- 1 Scrum Master
- 1 QA

🤖 The Codebase

index.html (with some js code)



cognizant softvision

This is the situation - 2nd iteration

👩🏻‍💼 Product Owner

Asks for a new increment of the application for the MVP.

the new specs are:

- It should save the number to DB
- It should be a SPA with a nice UI/UX touch

👨🏼‍💻 Developers

Team structure:

- 2 Ruby dev
- 2 React dev
- 1 Scrum Master
- 1 QA

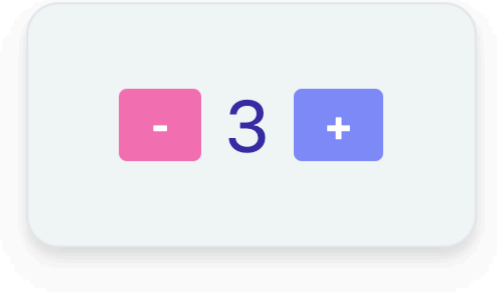
Teams of two for redundancy and code reviews.

🤖 Codebases

The initial index.html has been refactored 🤦 entirely.

A new Ruby project has been created.

A new React project has been created.



cognizant softvision

Here’s the Problem

The MVP web application is a success

PO: Let’s deliver it also as a native mobile real-time app… globally!

…

👨🏼‍💻 Is the architecture scalable?

👩🏻‍💻 Are new skills needed in the team?

👨🏼‍💼 Am I over the budget?

Here’s why it is hard to solve

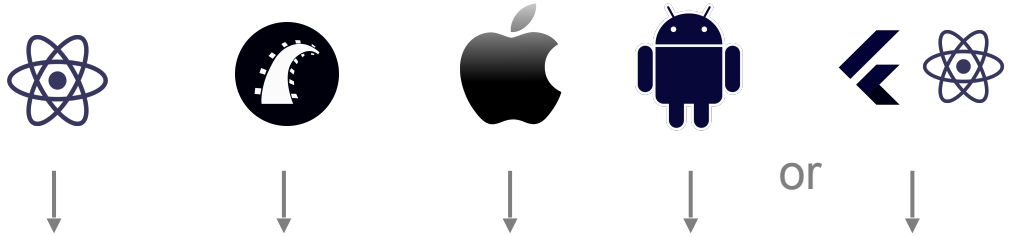
Budgets, time, devs & computers are limited resources

- Needed skills are hard to be find;
- Big teams are less productive than small ones;
- Human relations are hard to be established;
- Without adding extra complexity layers, the tech stack is not able to meet our new demands;
- Complex projects are hard to be maintained and prone to errors. That’s a fact;
- Consistency (both visual and functional), it’s another 🦣 in the room;
- Uncertainty is counterproductive;

...

Productivity killers

Tech stack:



Big teams:

web | API | iOS | android | cross

⚠️ 3 or 4 teams that do not know the business logic of the app; frequent syncs needed.

Orchestration:

- 👩🏻 How do we organize?
- 👨🏼‍💻 What/how are we building?
- 🕵🏻‍♀️ What are we testing?

Elixir Lang - Now & beyond

cognizant softvision

The Solution - Elixir

A mature, battle tested language with a large ecosystem

Various IT domains can be handled by this single language, which reduces the mixing of technologies as the application develops.

elixir

Domain	Project
Web Applications	Phoenix, LiveView
Machine learning & AI	NX Project
Native Applications	Phoenix, LiveView Native
Embedded/IoT	Nerves, Firefly

cognizant softvision

Short History

BEAM Languages

Elixir
Erlang
Gleam
Alpaca
...

ERLANG

BEAM
Virtual Machine

ERICSSON

2000s

1986

Elixir Lang - Now & beyond

cognizant softvision

Why Erlang has been created?

Demanding requirements of TelCo systems back in the 1980s:

- **Real-time** systems - Processes data with minimal delay.
- **Concurrent** systems - Multiple tasks running simultaneously.
- **Distributed** systems - Networked systems sharing resources.
- **Fault-tolerance -** Ability to continue functioning after failures.

Limitations of Existing Programming Languages

Inefficient handling of concurrency
Poor support for distribution – client/ server
Difficulties with fault-tolerance

Key Features of Erlang

Lightweight processes (BEAM process)
Message-passing concurrency (Actor/ Model)
Fault-tolerance
Distributed – run on multiple nodes
Hot code swapping - Fix while running

Why Elixir has been created?

Elixir was created to provide developers with a modern, productive, and scalable language for building distributed and fault-tolerant systems.

Demanding requirements of systems back in the 1980s ...

- **Real-time** systems - Processes data with minimal delay.
- **Concurrent** systems - Multiple tasks running simultaneously.
- **Distributed** systems - Networked systems sharing resources.
- **Fault-tolerance -** Ability to continue functioning after failures.

… are the same NOW!

cognizant softvision

Companies using Elixir

They migrated from Rails to Phoenix and reduced app servers from 150 to just eight. They also reduced their team size as they no longer required as many developers to support their ongoing needs.

cognizant softvision

The Solution - Elixir

Tech stack:

Elixir & Phoenix

One team:

Cross Platform - **truly native**

A single codebase, singular business logic implementation.

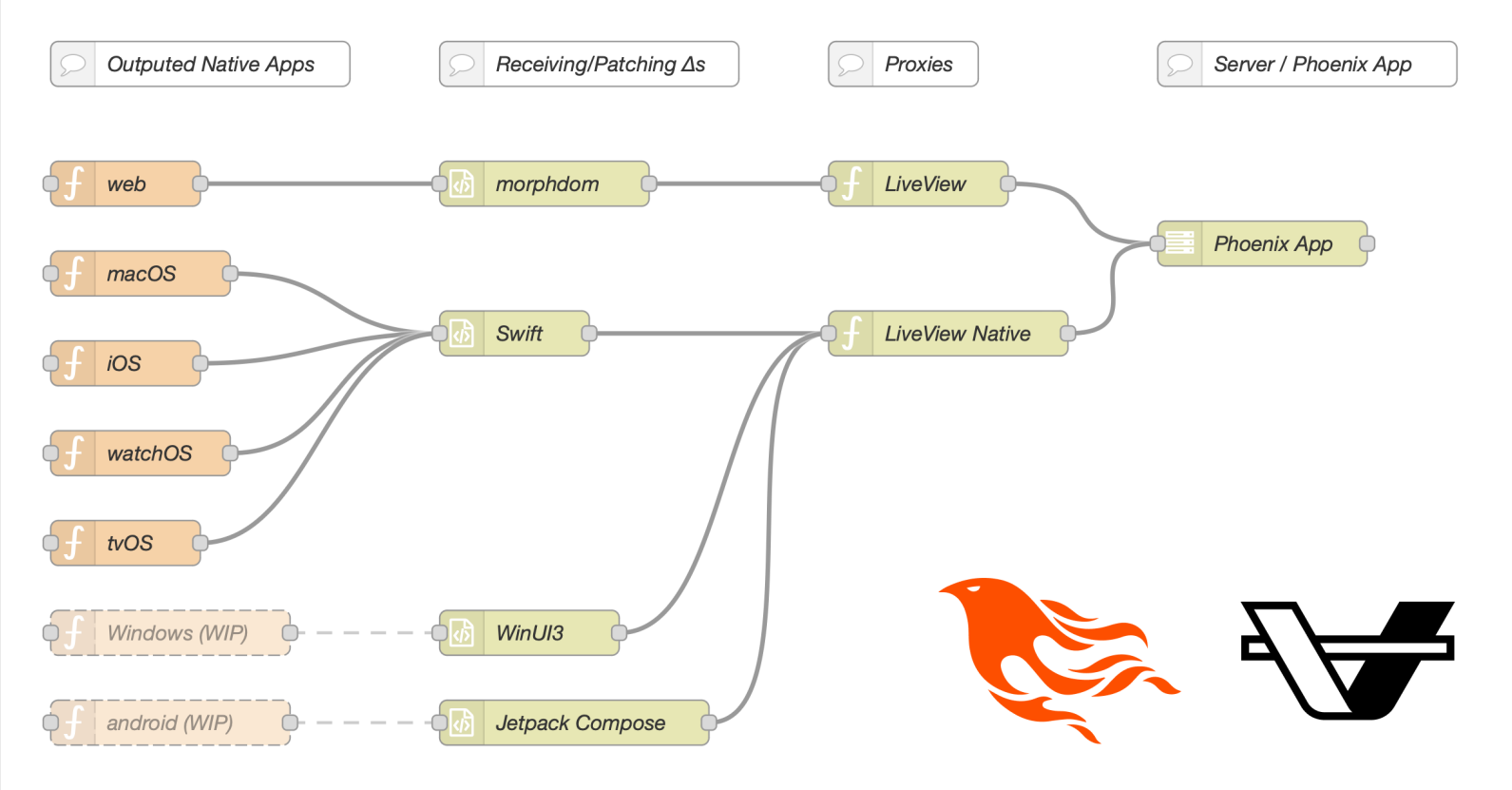
Orchestration:

- We know what to deliver.
- We know what to build.
- We know what to test.

Elixir Lang - Now & beyond

cognizant softvision

LiveView/LiveView Native



Elixir Lang - Now & beyond

cognizant softvision

Project management

Request: Increase title size

Acceptance criteria: Expect the title size to be bigger across the apps.

Mobile

- [iOS] - increase title size
- [iOS] - release app
- [android] - increase title size
- [android] - release app

Web

- increase title size
- deploy

VS

Shared

- increase title size
- deploy

iOS specific

- no tasks

android specific

- no tasks

specific Targets a specifc platform matter.

Costs & Consistency

CI/CD Costs

Less builds/less money

Consistent release

No need to wait for AppStore/ Google Play

App Store

Google play

Elixir Lang - Now & beyond

cognizant softvision

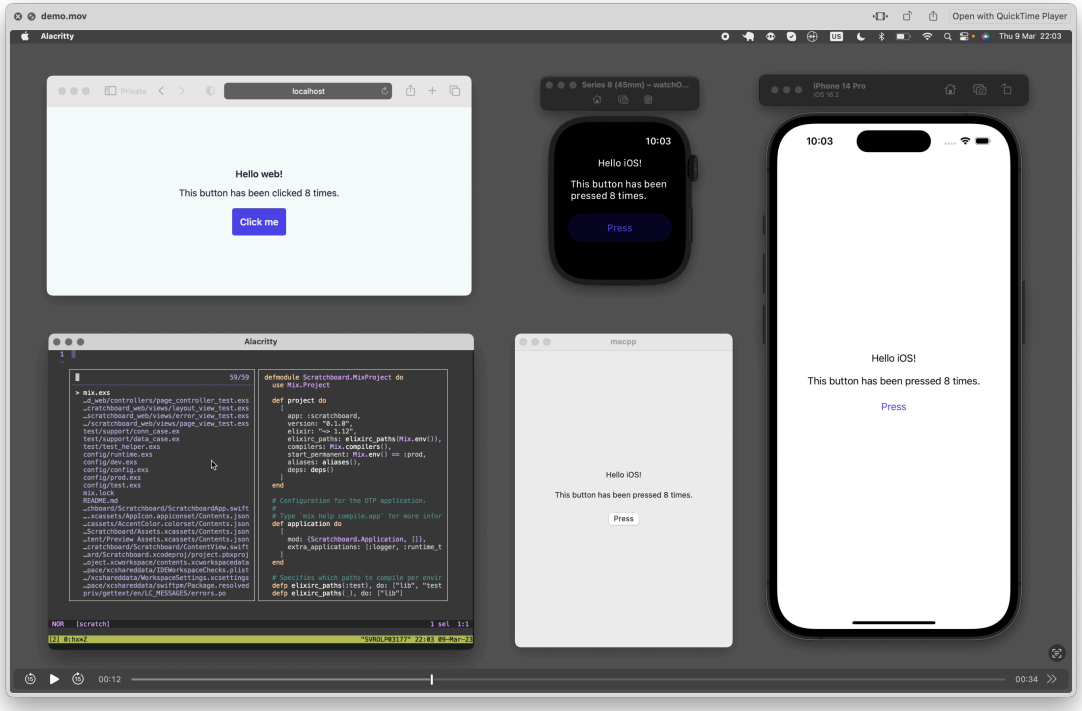
Demo

Presentation:

https://youtu.be/mRf20AANdYk

Demo

LiveView web app with LiveView Native MacOs, WatchOS and iOS apps. Rendered via a single Phoenix app



cognizant softvision

cognizant softvision

Thank you

Constantin Angheloiu

constantin.angheloiu@gmail.com
@cmnstmntmn