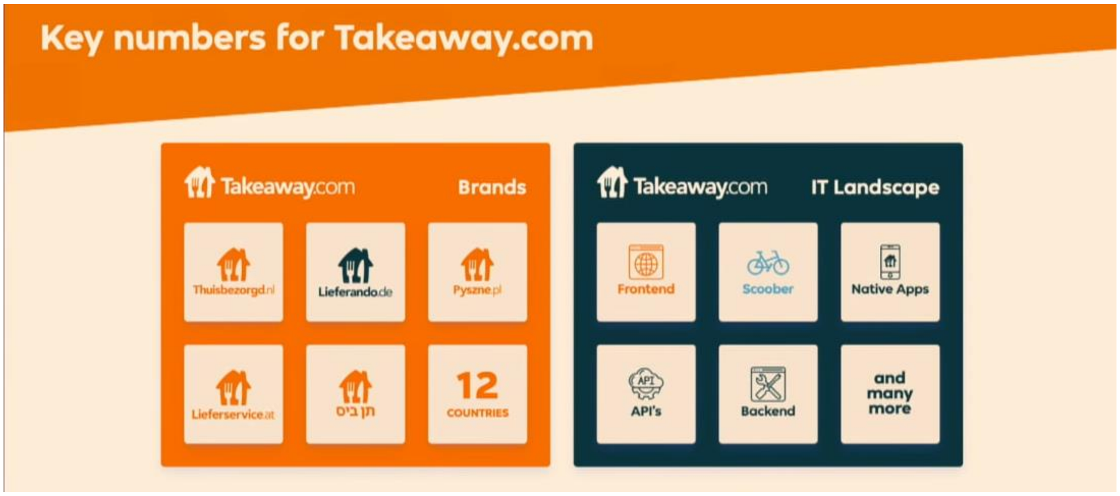


Key numbers for Takeaway.com

Takeaway.com Brands

Thuisbezorgd.nl

Lieferando.de

Pyszne.pl

Lieferservice.at

תן ביס

12 COUNTRIES

Takeaway.com IT Landscape

Frontend

Scoober

Native Apps

API's

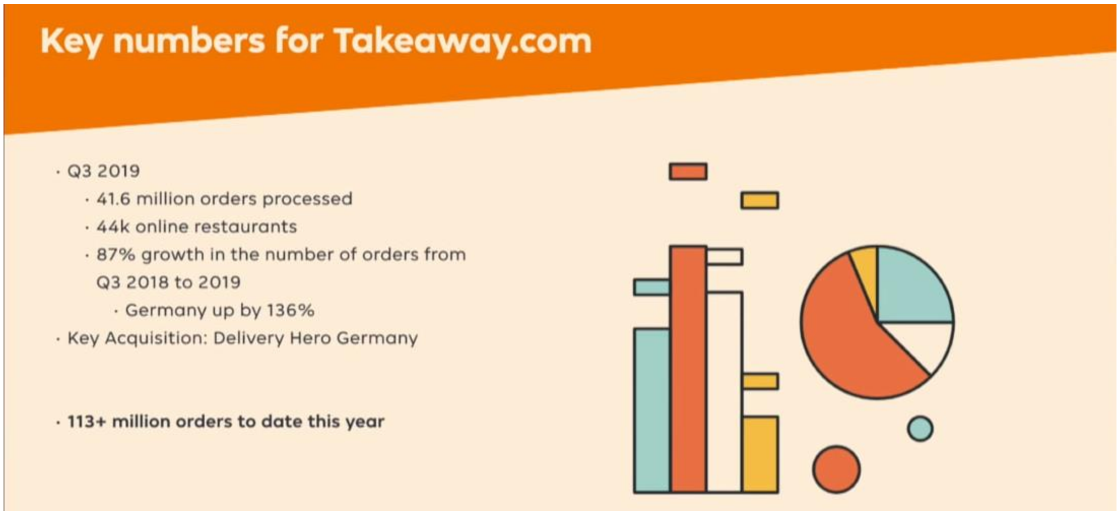
Backend

and many more

Takeaway.com has been in business for 20 years. We support 150M orders every year and we operate in more than 10 countries. Over time, business requirements change, new technologies arise and certain processes become outdated. Combined with having development teams in offices all over the world, you can imagine the difficulties that a business can face while scaling their software architecture. This talk provides general guidelines and strategies to plan and implement future-proof generations of APIs. It tells the story of how the development teams at Takeaway.com are tackling these challenges.

In the first part of the talk, Michele will speak about the Logistics team and how they approach the complexity of handling more than 10.000 drivers worldwide. He will discuss the principles behind building their APIs such as Domain Driven Design (DDD) and Microservices.

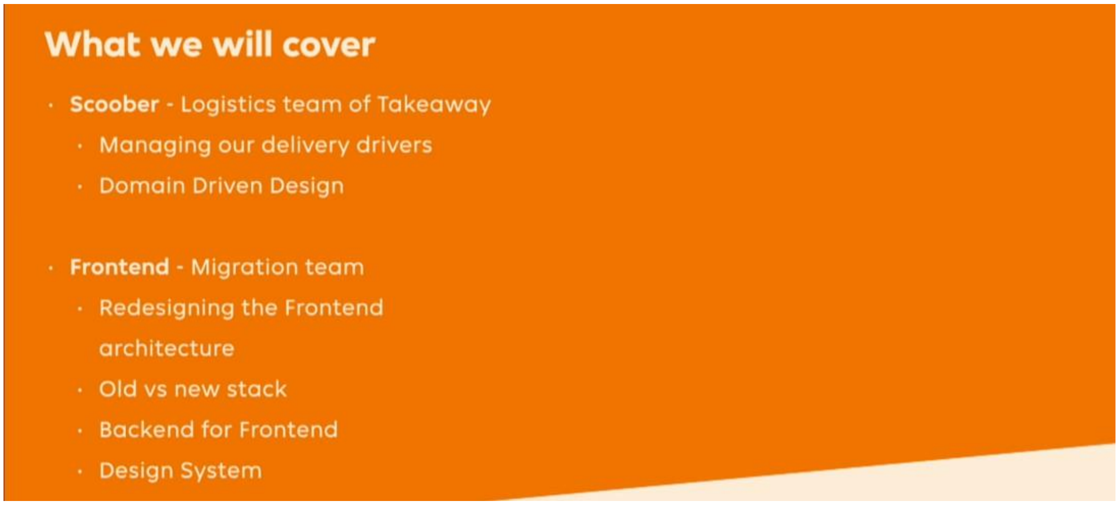
Then, Matt will talk about the current migration project for rebuilding the front-end application at Takeaway.com. He will walk through the evolution of their 20-year-old codebase, and why the front-end team decided to migrate to a more modern stack while implementing best practices for scaling the front end, such as Backend for Frontend (BFF) and Server-Side Rendering (SSR) [...]



Key numbers for Takeaway.com

- Q3 2019
  - 41.6 million orders processed
  - 44k online restaurants
  - 87% growth in the number of orders from Q3 2018 to 2019
    - Germany up by 136%
- Key Acquisition: Delivery Hero Germany

- **113+ million orders to date this year**

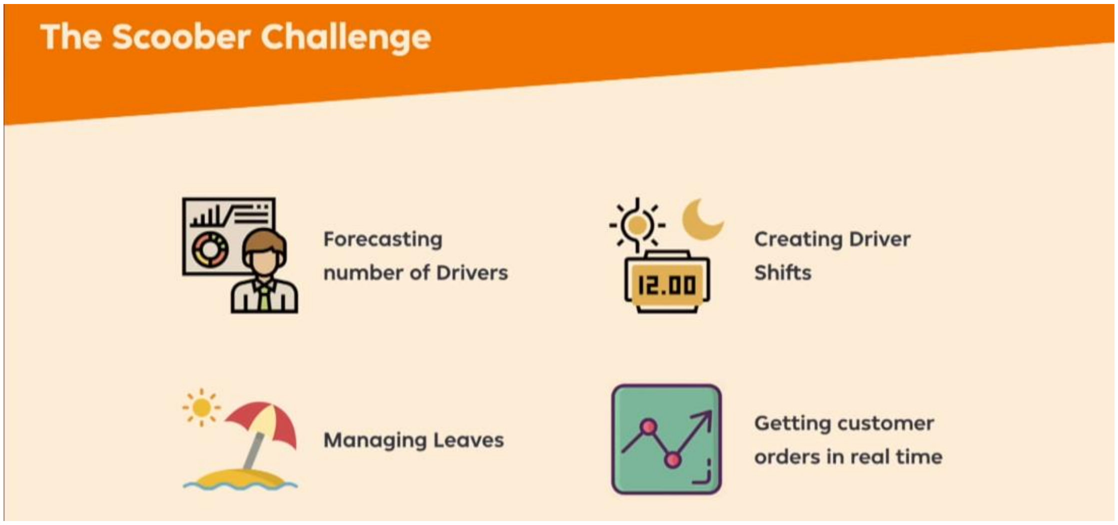


What we will cover

- **Scoober** - Logistics team of Takeaway
  - Managing our delivery drivers
  - Domain Driven Design
- **Frontend** - Migration team
  - Redesigning the Frontend architecture
  - Old vs new stack
  - Backend for Frontend
  - Design System

Scoober

Scoober is our delivery service for restaurants using our own employed drivers



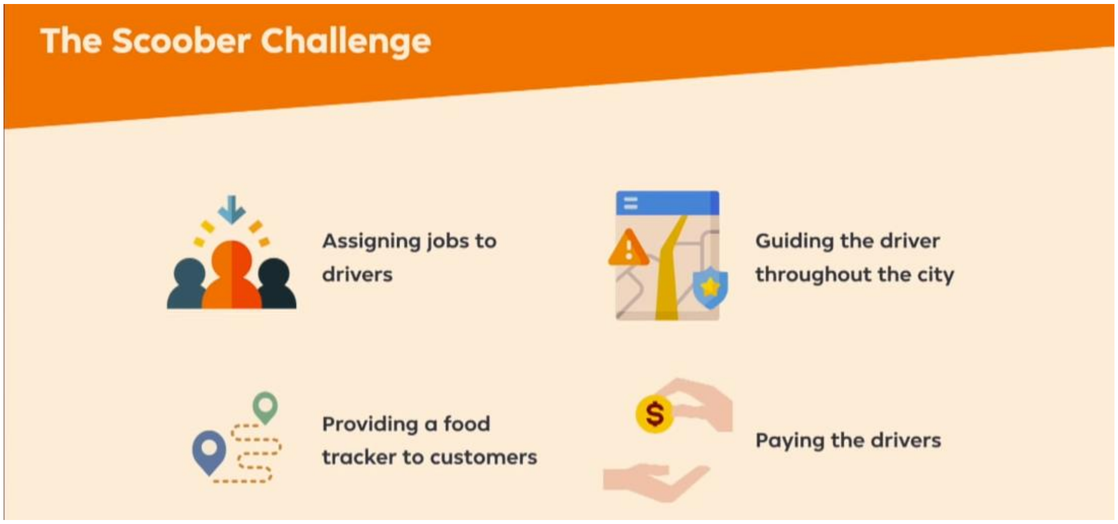
The Scoober Challenge

Forecasting number of Drivers

Creating Driver Shifts

Managing Leaves

Getting customer orders in real time



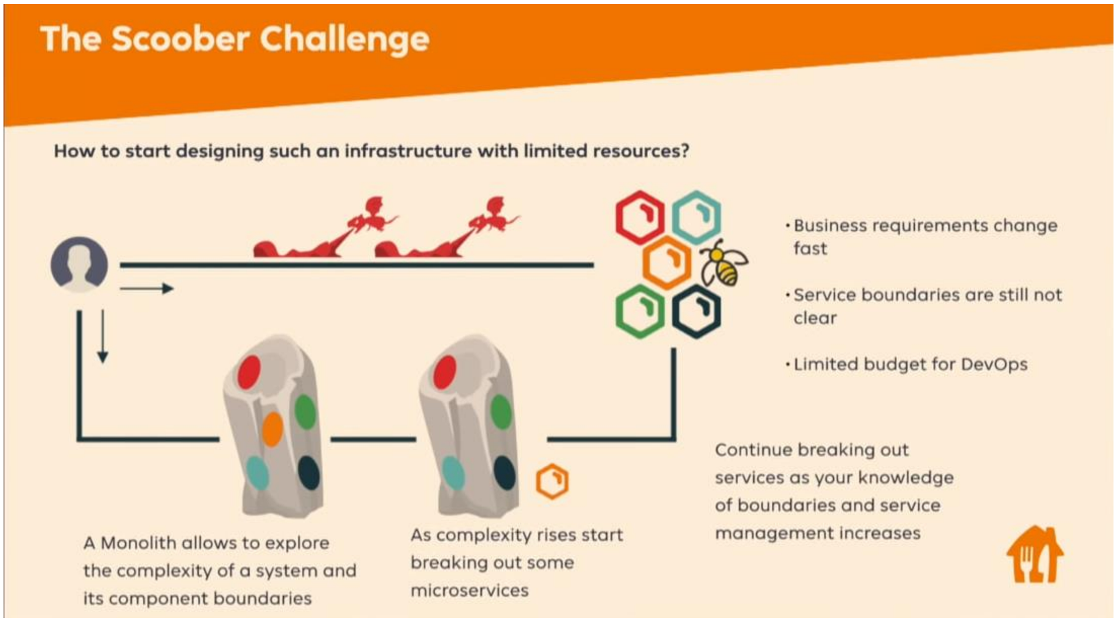
The Scoober Challenge

Assigning jobs to drivers

Guiding the driver throughout the city

Providing a food tracker to customers

Paying the drivers



The Scoober Challenge

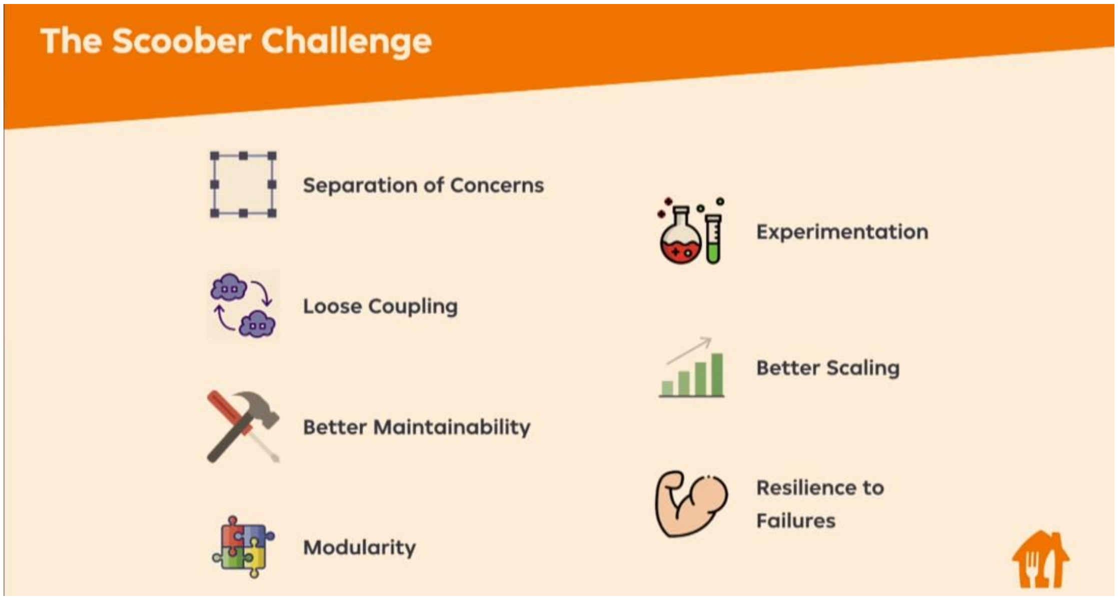
How to start designing such an infrastructure with limited resources?

- Business requirements change fast
- Service boundaries are still not clear
- Limited budget for DevOps

A Monolith allows to explore the complexity of a system and its component boundaries

As complexity rises start breaking out some microservices

Continue breaking out services as your knowledge of boundaries and service management increases



The Scoober Challenge

Separation of Concerns

Loose Coupling

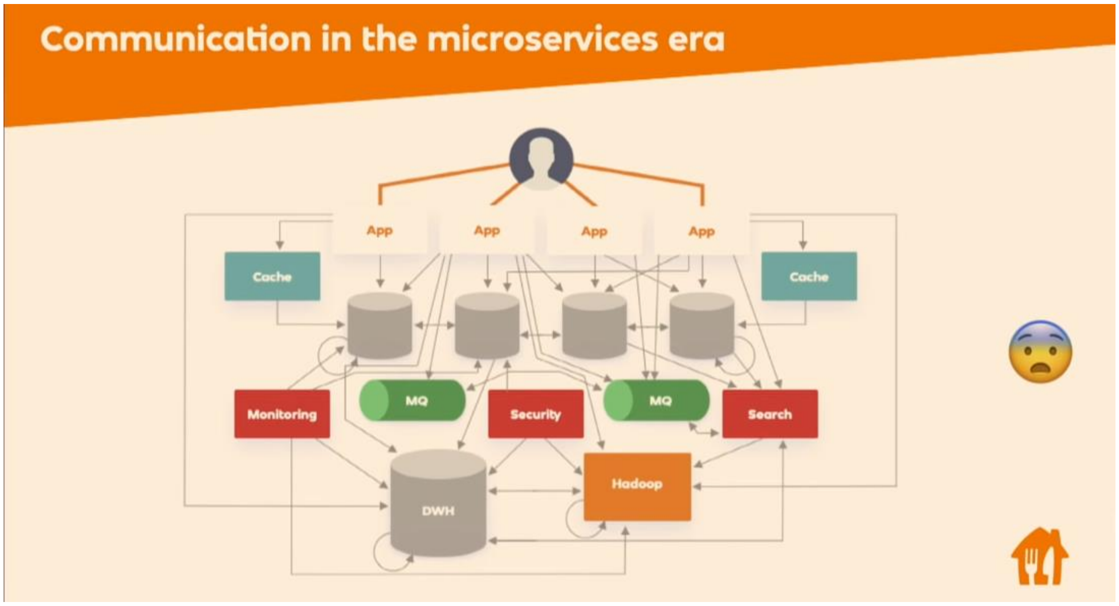
Better Maintainability

Modularity

Experimentation

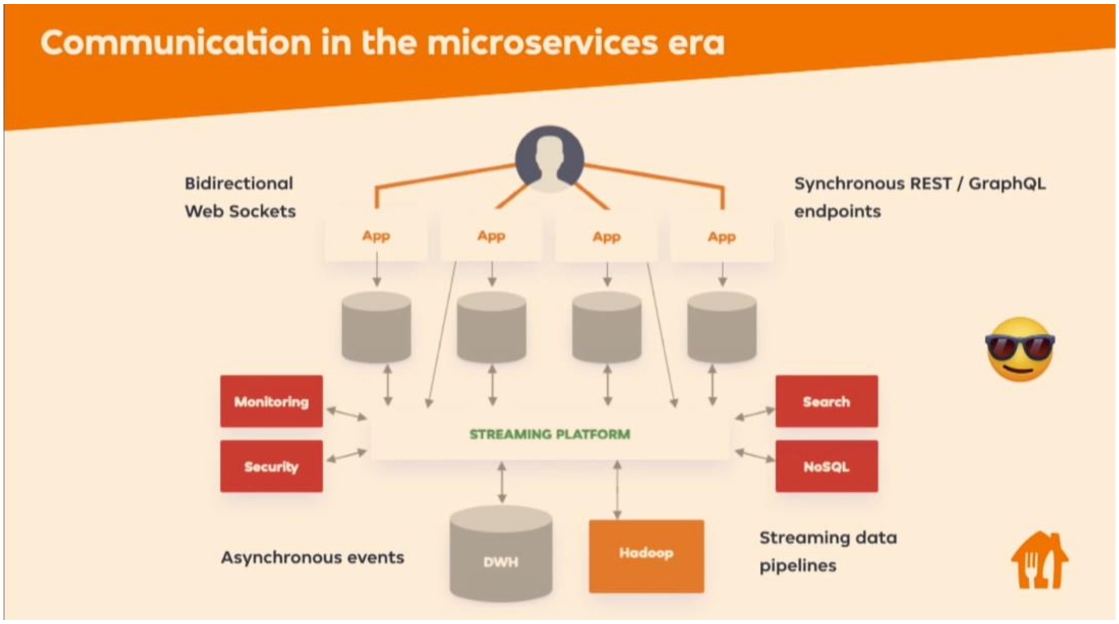
Better Scaling

Resilience to Failures



Communication in the microservices era

We need to move from the above architecture



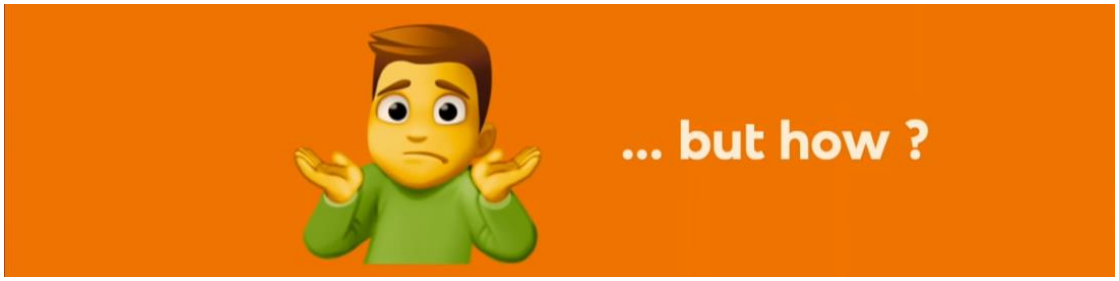
Communication in the microservices era

Bidirectional Web Sockets

Synchronous REST / GraphQL endpoints

Asynchronous events

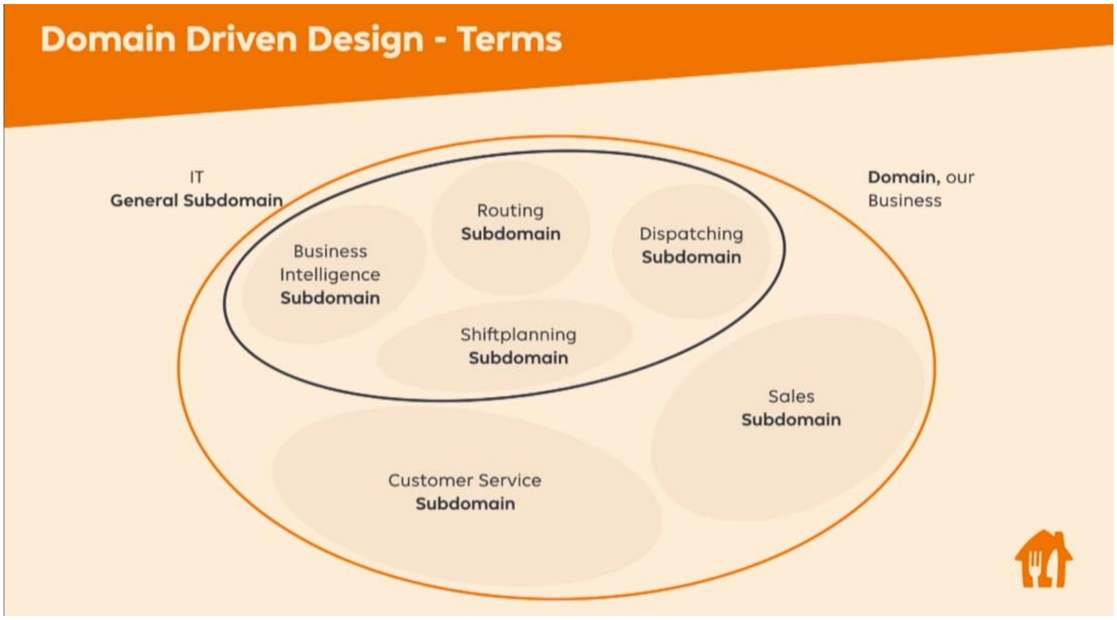
Streaming data pipelines



... but how ?

Domain Driven Design

Domain-driven design (DDD) is an approach to software development for complex needs by connecting the implementation to an evolving model.



Domain Driven Design - Terms

IT
General Subdomain

Domain, our
Business

Business
Intelligence
Subdomain

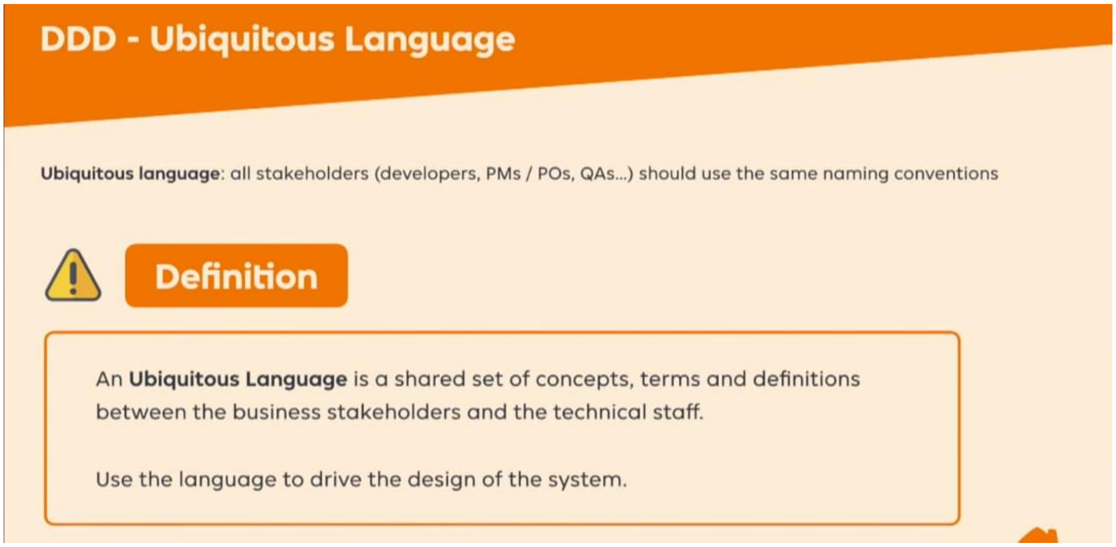
Routing
Subdomain

Dispatching
Subdomain

Shiftplanning
Subdomain

Sales
Subdomain

Customer Service
Subdomain



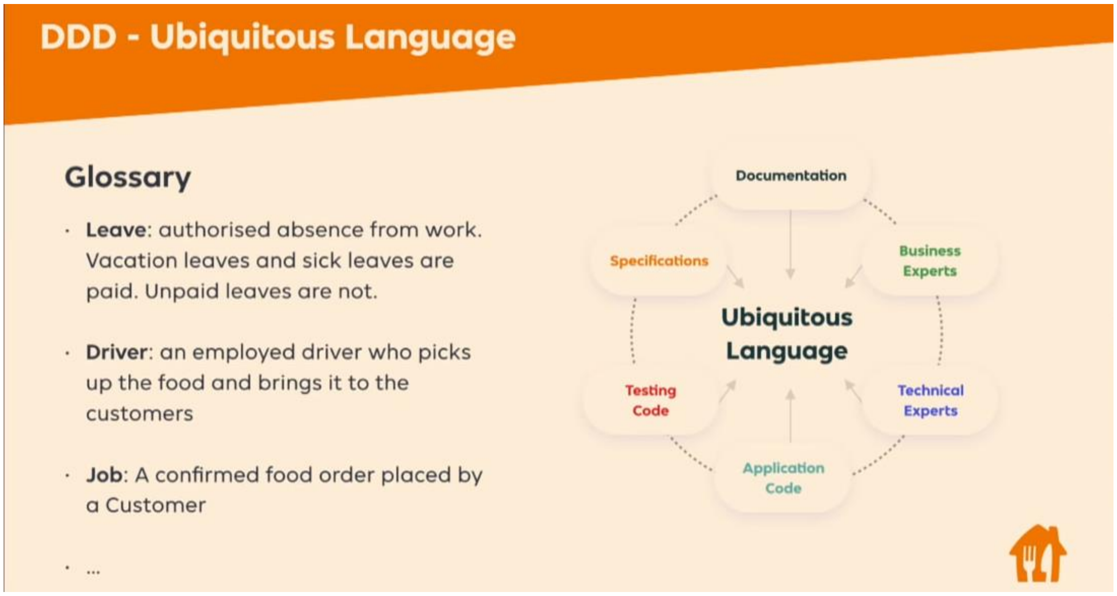
DDD - Ubiquitous Language

Ubiquitous language: all stakeholders (developers, PMs / POs, QAs...) should use the same naming conventions

⚠ **Definition**

An **Ubiquitous Language** is a shared set of concepts, terms and definitions between the business stakeholders and the technical staff.

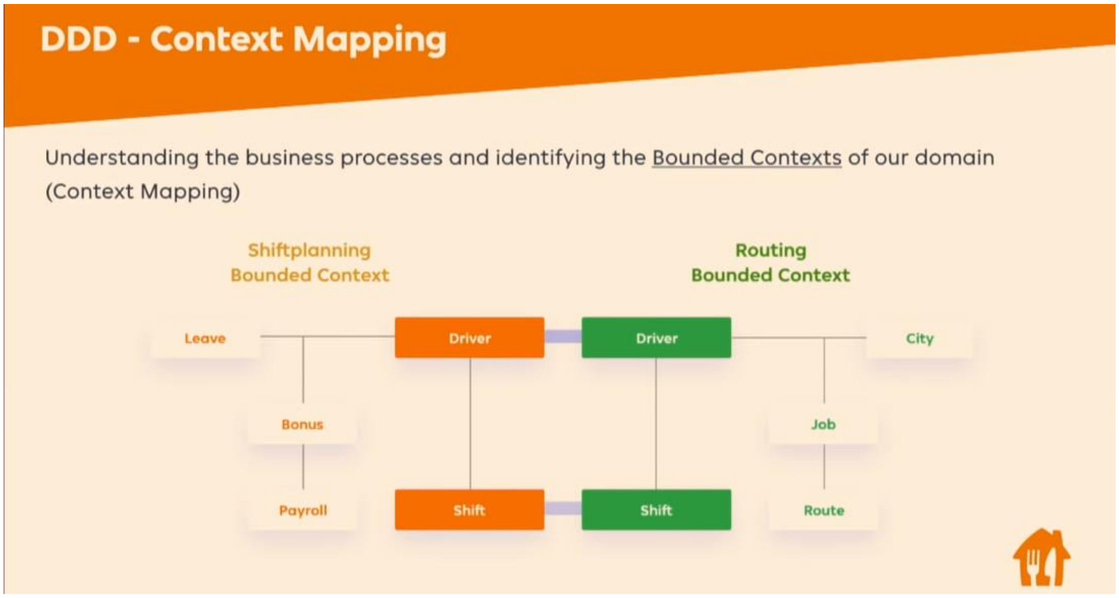
Use the language to drive the design of the system.



DDD - Ubiquitous Language

Glossary

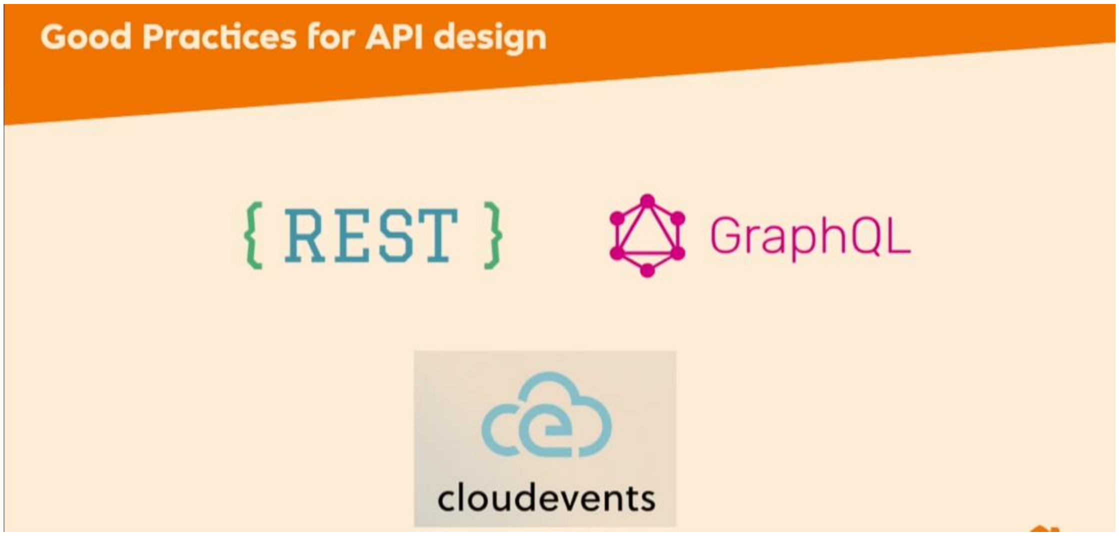
- **Leave**: authorised absence from work. Vacation leaves and sick leaves are paid. Unpaid leaves are not.
- **Driver**: an employed driver who picks up the food and brings it to the customers
- **Job**: A confirmed food order placed by a Customer
- ...



DDD - Context Mapping

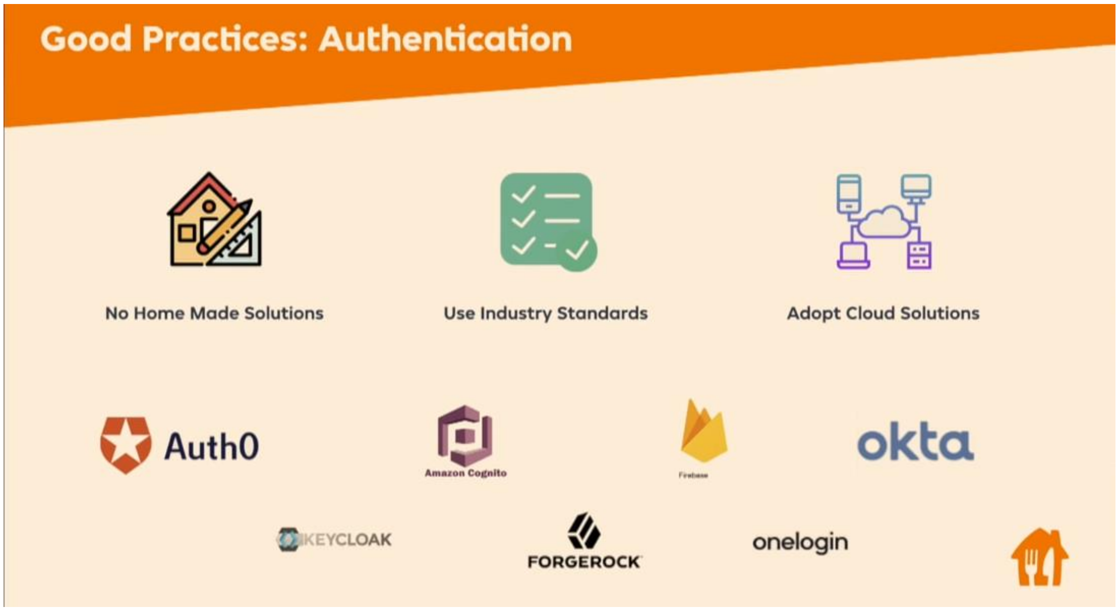
Understanding the business processes and identifying the Bounded Contexts of our domain (Context Mapping)

we identify the main entities in a bounded context/domain and the relationships between the different contexts.



Good Practices for API design

{ REST }
GraphQL
cloudevents



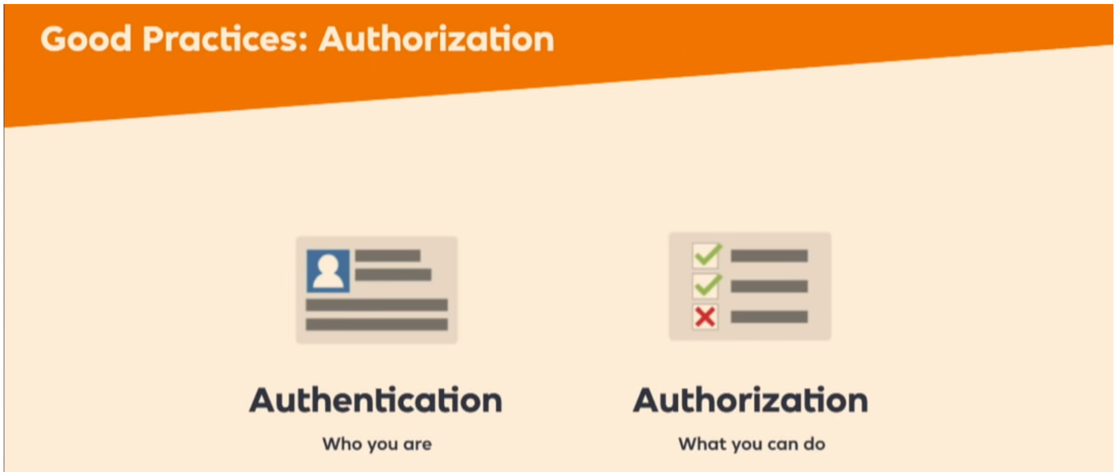
Good Practices: Authentication

No Home Made Solutions

Use Industry Standards

Adopt Cloud Solutions

Auth0
Amazon Cognito
Firebase
okta
KEYCLOAK
FORGEROCK
onelogin



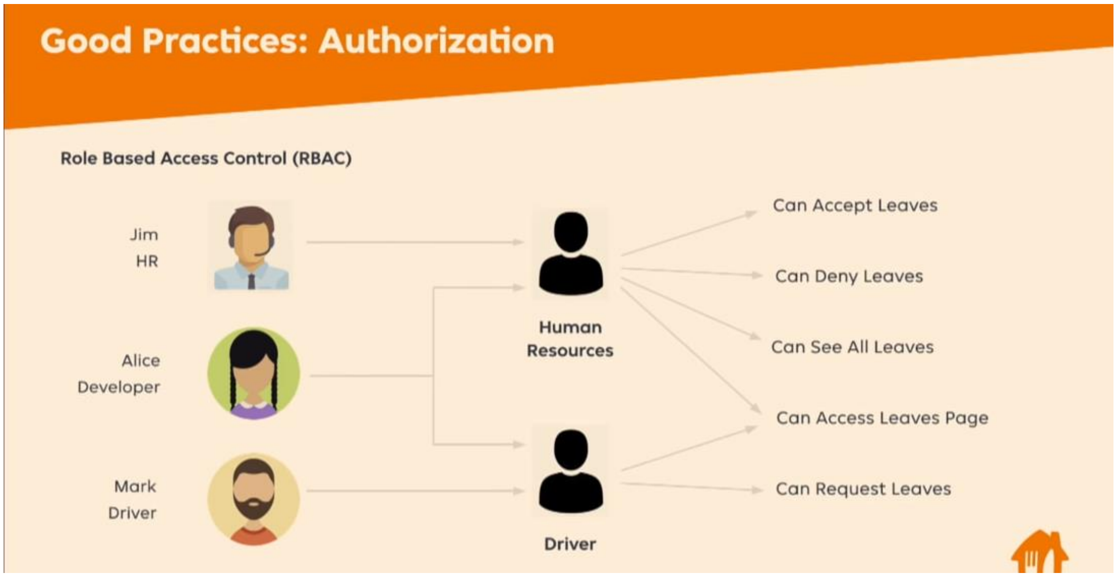
Good Practices: Authorization

Authentication

Who you are

Authorization

What you can do



Good Practices: Authorization

Role Based Access Control (RBAC)

Jim
HR

Alice
Developer

Mark
Driver

Human Resources

Driver

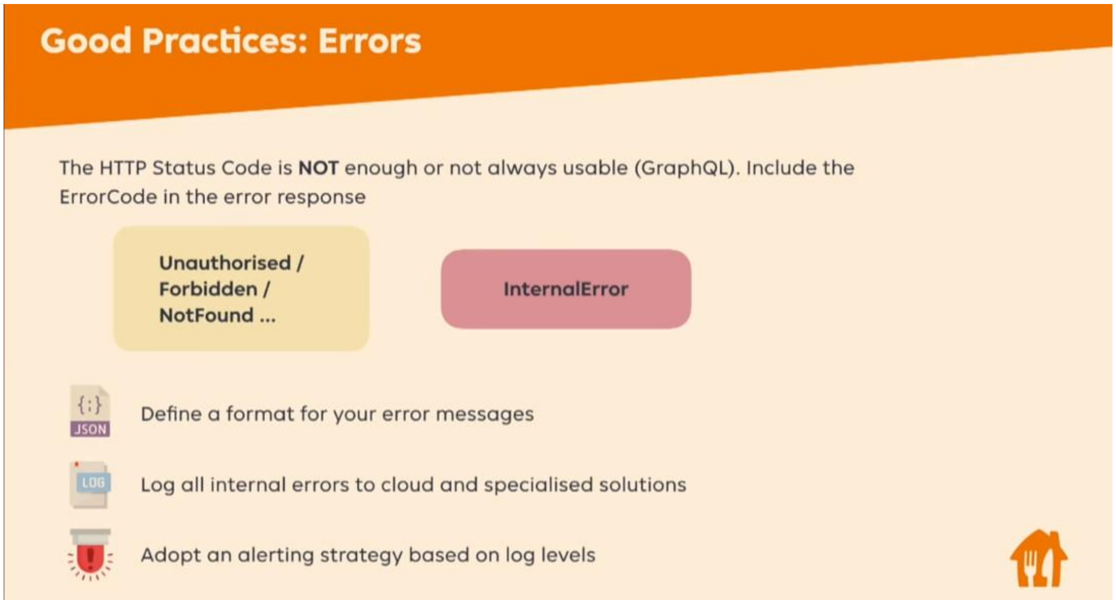
Can Accept Leaves

Can Deny Leaves

Can See All Leaves

Can Access Leaves Page

Can Request Leaves



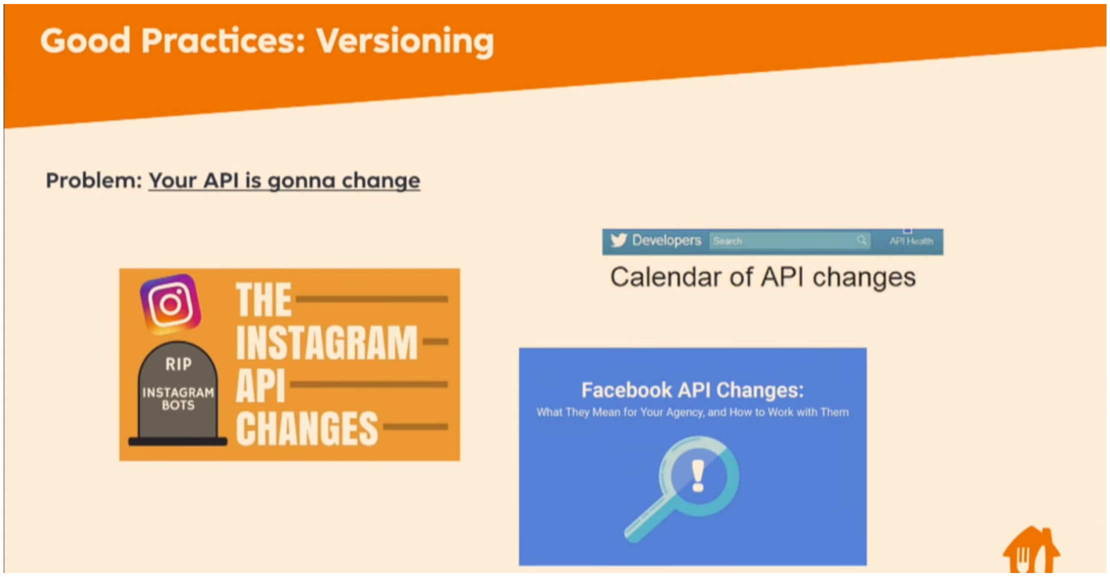
Good Practices: Errors

The HTTP Status Code is **NOT** enough or not always usable (GraphQL). Include the ErrorCode in the error response

Unauthorised / Forbidden / NotFound ...

InternalError

- Define a format for your error messages
- Log all internal errors to cloud and specialised solutions
- Adopt an alerting strategy based on log levels



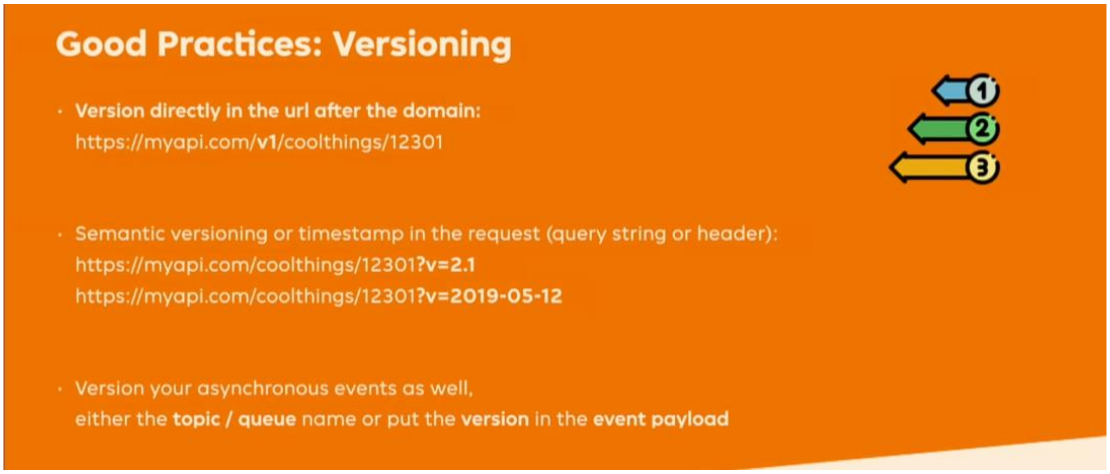
Good Practices: Versioning

Problem: Your API is gonna change

Calendar of API changes

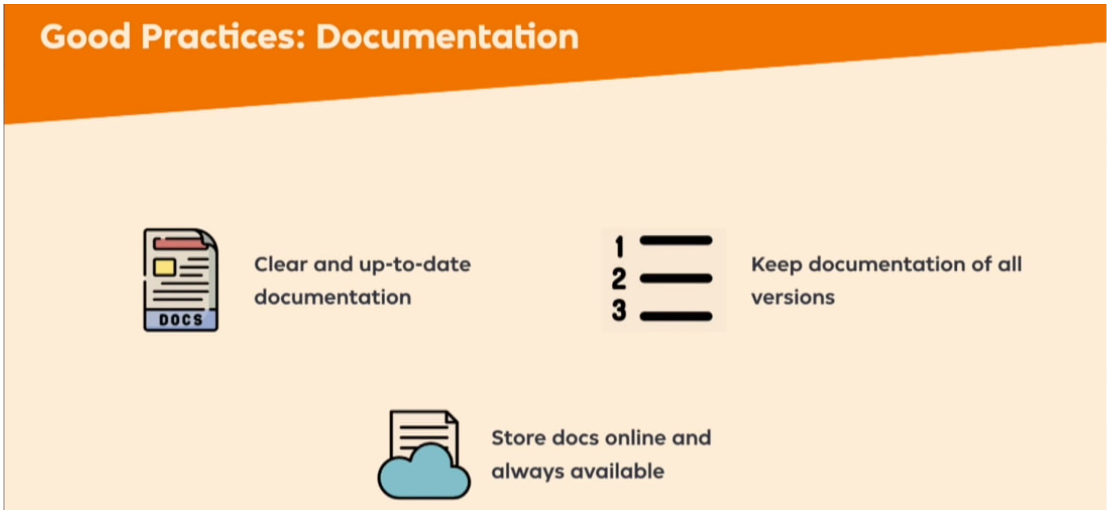
Facebook API Changes:

What They Mean for Your Agency, and How to Work with Them



Good Practices: Versioning

- **Version directly in the url after the domain:**
  https://myapi.com/**v1**/coolthings/12301
- Semantic versioning or timestamp in the request (query string or header):
  https://myapi.com/coolthings/12301**?v=2.1**
  https://myapi.com/coolthings/12301**?v=2019-05-12**
- Version your asynchronous events as well,
  either the **topic / queue** name or put the **version** in the **event payload**



Good Practices: Documentation

Clear and up-to-date documentation

Keep documentation of all versions

Store docs online and always available

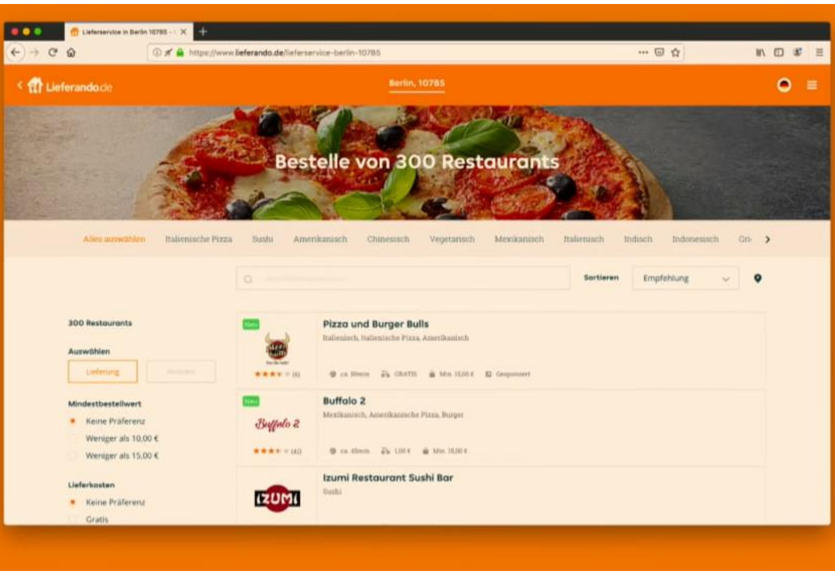
Good Practices: Testing

Use different environments

Blue / Green deployments

Test Automation

Frontend



Bestelle von 300 Restaurants

300 Restaurants

Auswählen

Mindestbestellwert
● Keine Präferenz
○ Weniger als 10,00 €
○ Weniger als 15,00 €

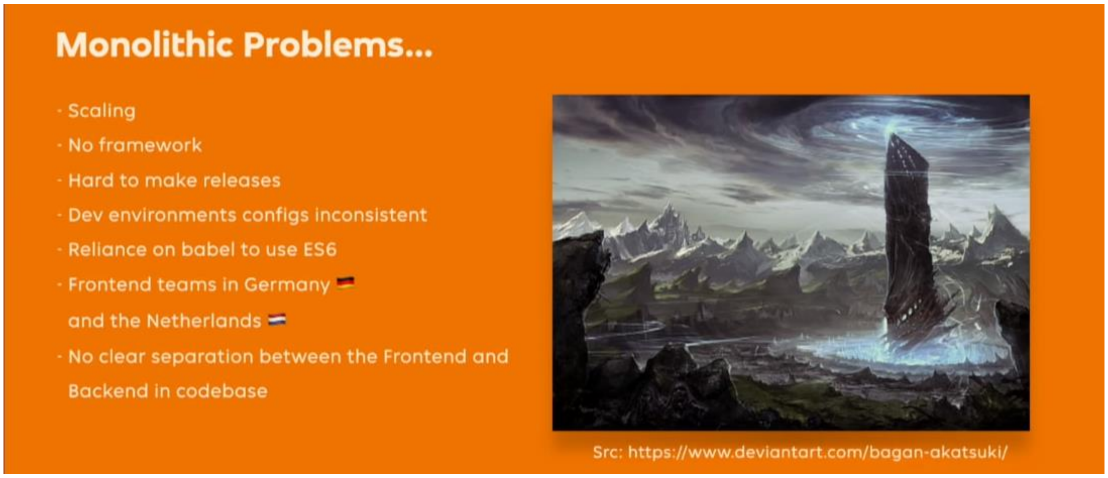
Lieferkosten
● Keine Präferenz
○ Gratis

Pizza und Burger Bulls
Italienisch, Italienische Pizza, Amerikanisch
★★★★☆ (6) ca. 30min GRATIS Min. 15,00 € Gesponsert

Buffalo 2
Mexikanisch, Amerikanische Pizza, Burger
★★★★☆ (41) ca. 45min 1,00 € Min. 15,00 €

Izumi Restaurant Sushi Bar
Sushi

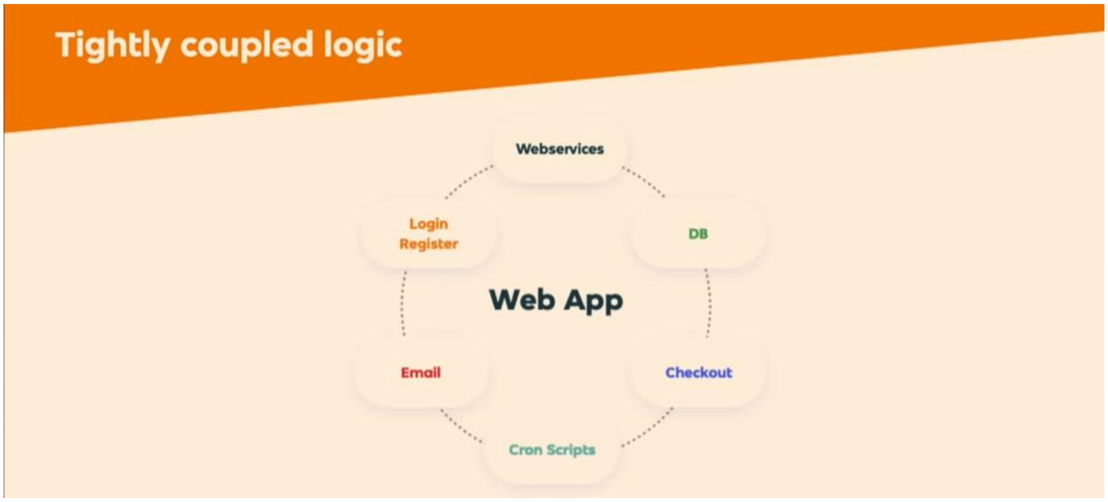
Our Current Stack



Monolithic Problems...

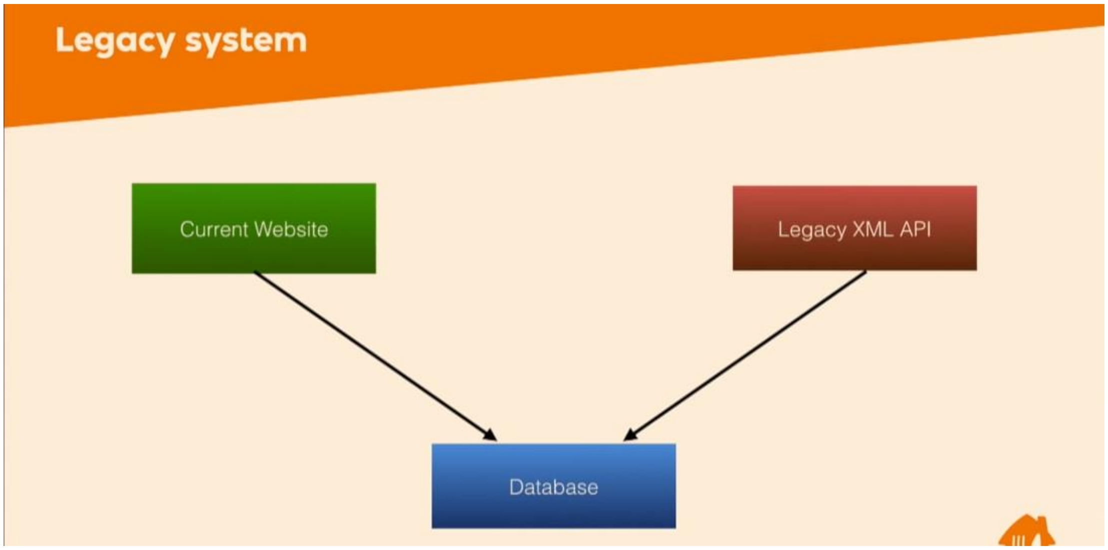
- Scaling
- No framework
- Hard to make releases
- Dev environments configs inconsistent
- Reliance on babel to use ES6
- Frontend teams in Germany 🇩🇪 and the Netherlands 🇳🇱
- No clear separation between the Frontend and Backend in codebase

Src: https://www.deviantart.com/bagan-akatsuki/

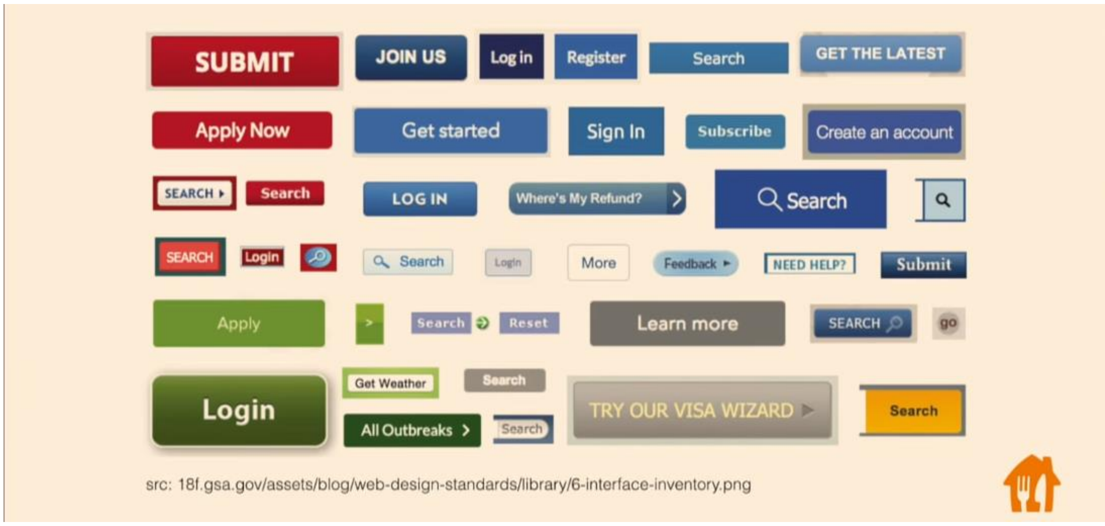


Tightly coupled logic

Developer Wack-a-mole

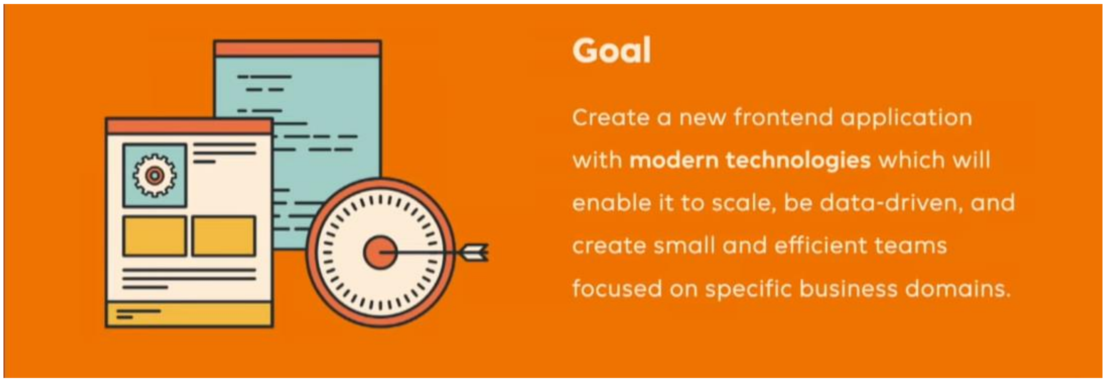


Legacy system



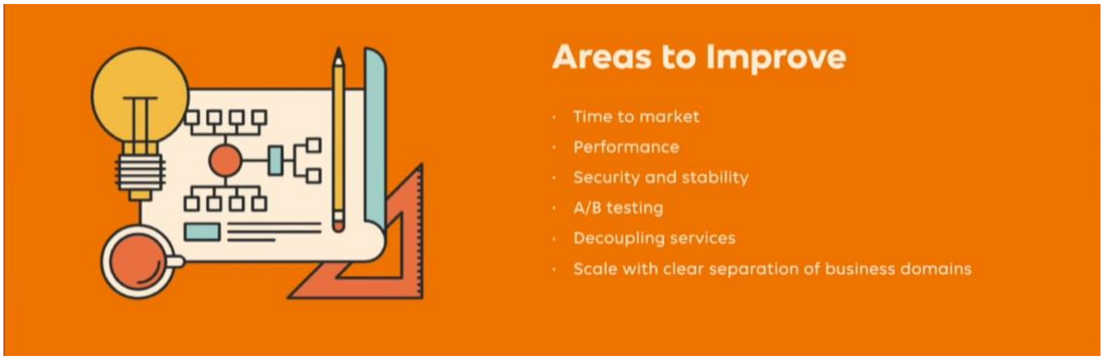
src: 18f.gsa.gov/assets/blog/web-design-standards/library/6-interface-inventory.png

Consumer Web: The Great Migration



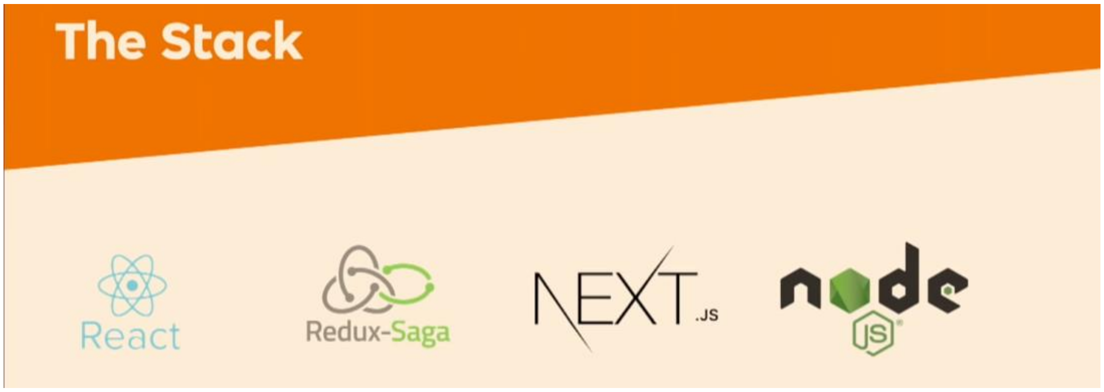
Goal

Create a new frontend application with **modern technologies** which will enable it to scale, be data-driven, and create small and efficient teams focused on specific business domains.



Areas to Improve

- Time to market
- Performance
- Security and stability
- A/B testing
- Decoupling services
- Scale with clear separation of business domains

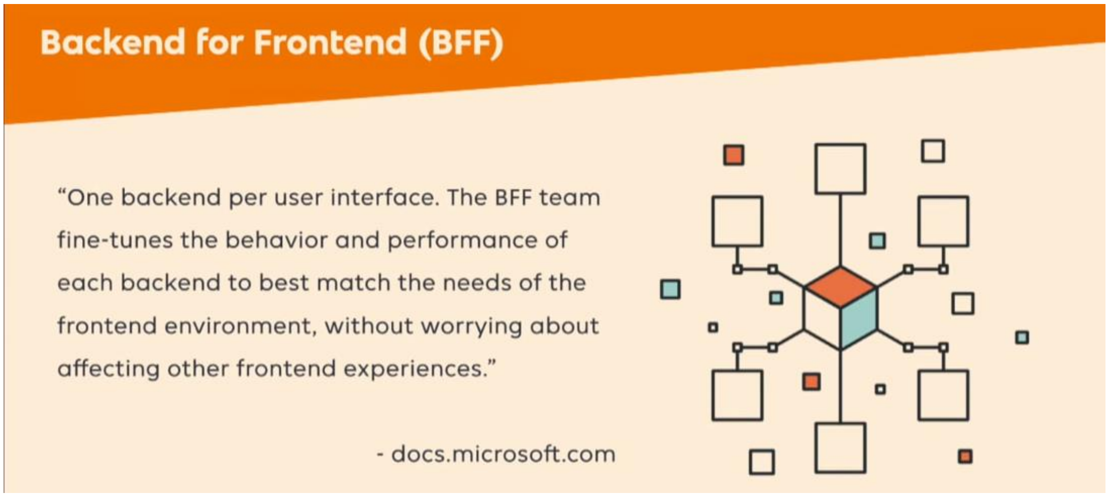


The Stack

React
Redux-Saga
NEXT.JS
node JS

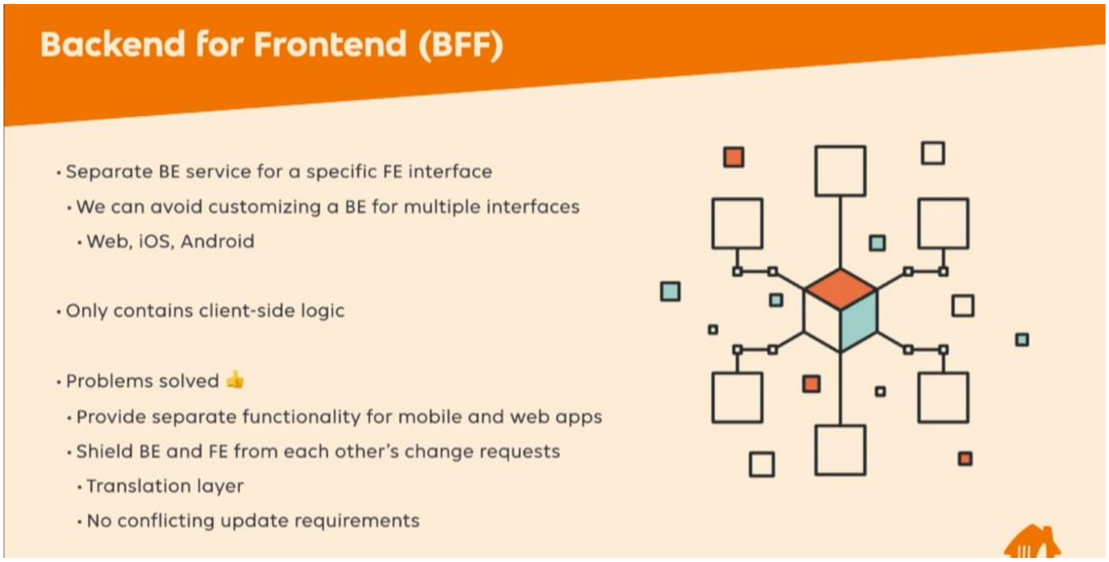
React and **Redux-Saga** on the FE, Sagas allows us to organize the side effects within our application clearly and organized. NextJS and NodeJS on the BE, **NextJS** gives us all the benefits from server-side rendering for improved SEO, isomorphic JS code and faster load times.

What about the Legacy XML API?



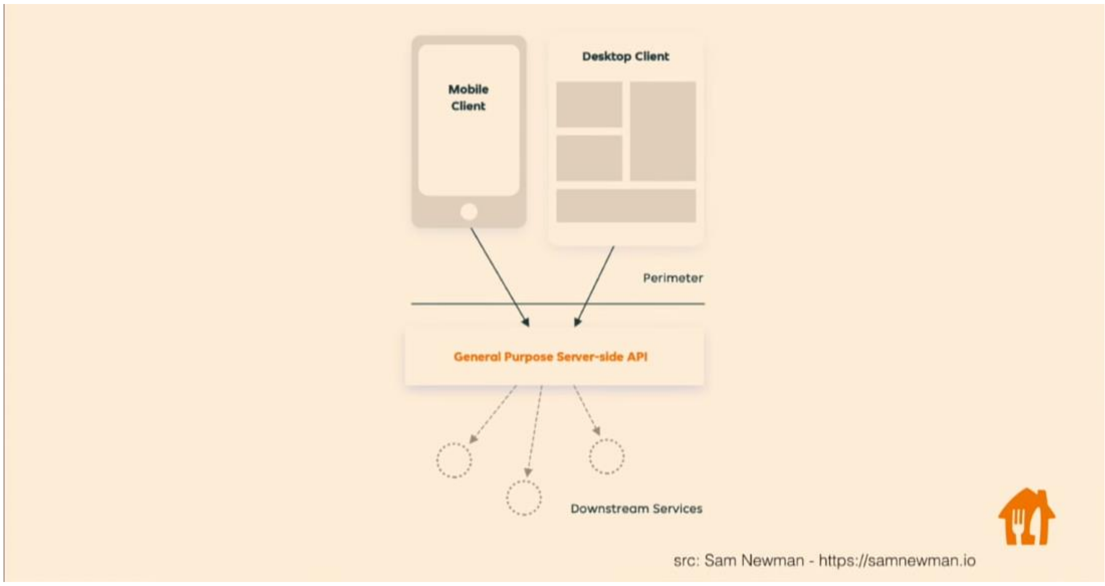
Backend for Frontend (BFF)

> “One backend per user interface. The BFF team fine-tunes the behavior and performance of each backend to best match the needs of the frontend environment, without worrying about affecting other frontend experiences.”
>
> - docs.microsoft.com

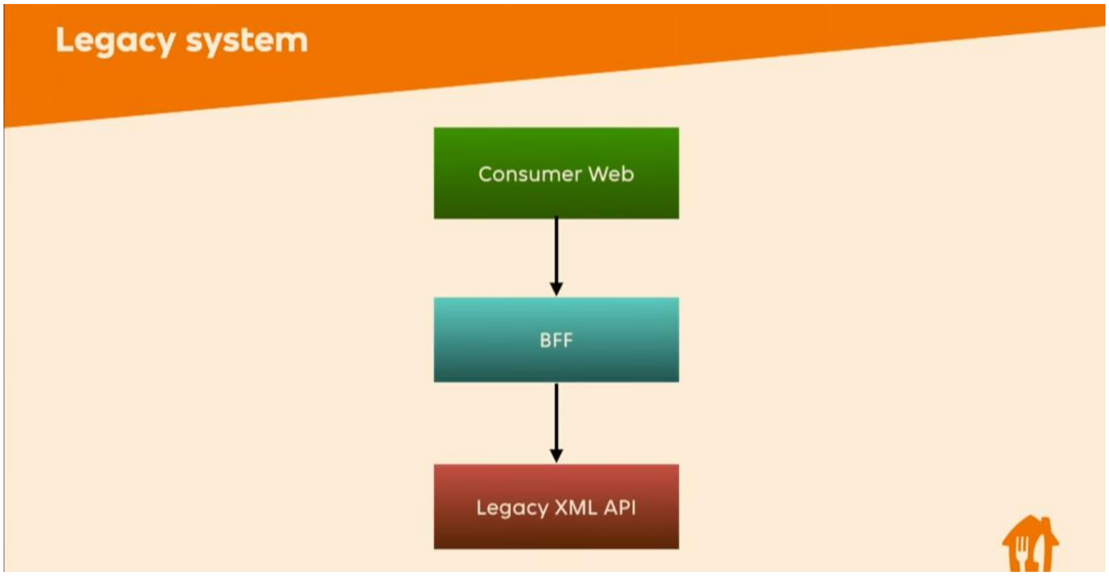


Backend for Frontend (BFF)

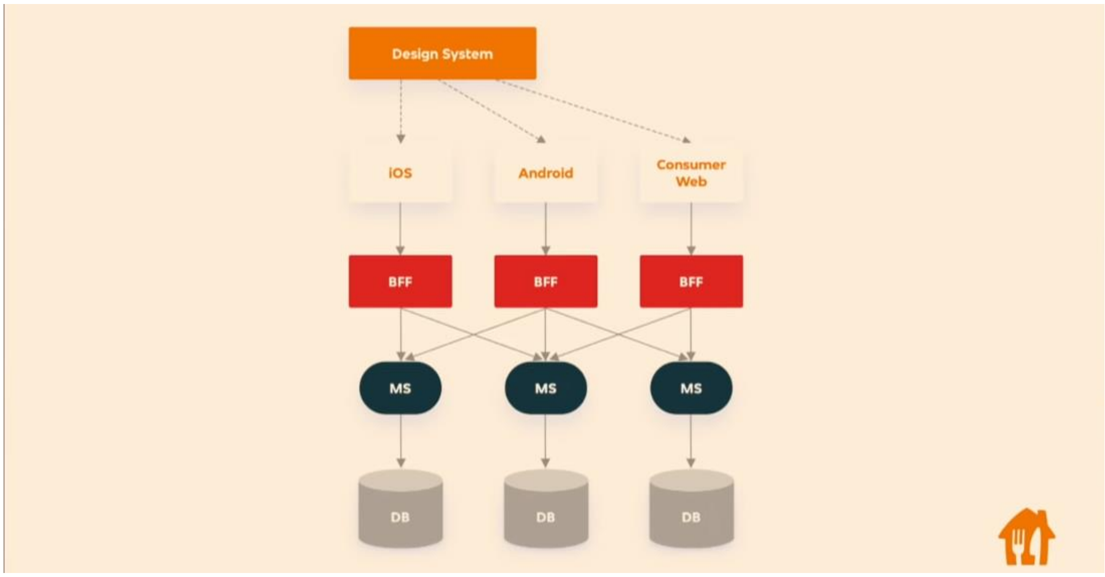
- Separate BE service for a specific FE interface
  - We can avoid customizing a BE for multiple interfaces
    - Web, iOS, Android
- Only contains client-side logic
- Problems solved 👍
  - Provide separate functionality for mobile and web apps
  - Shield BE and FE from each other’s change requests
    - Translation layer
    - No conflicting update requirements

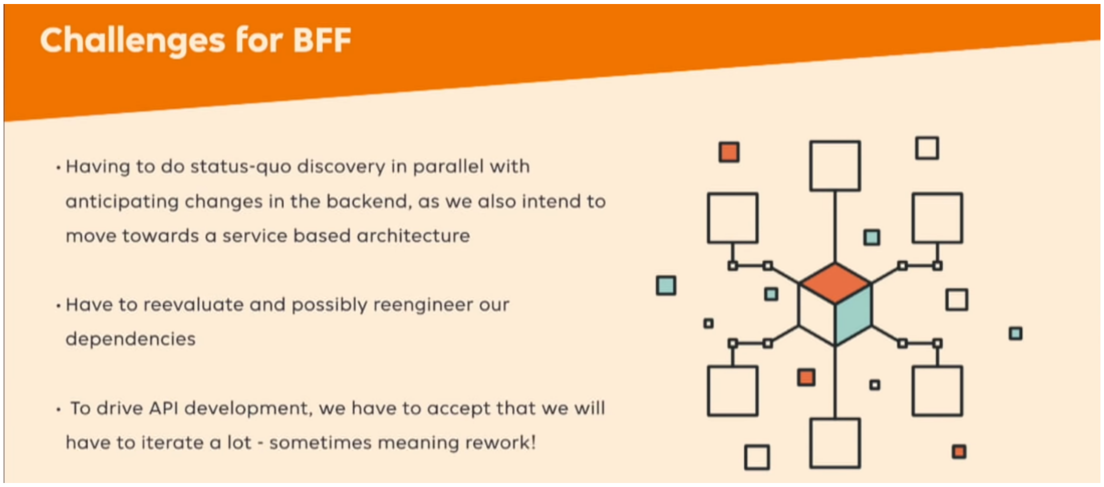


src: Sam Newman - https://samnewman.io



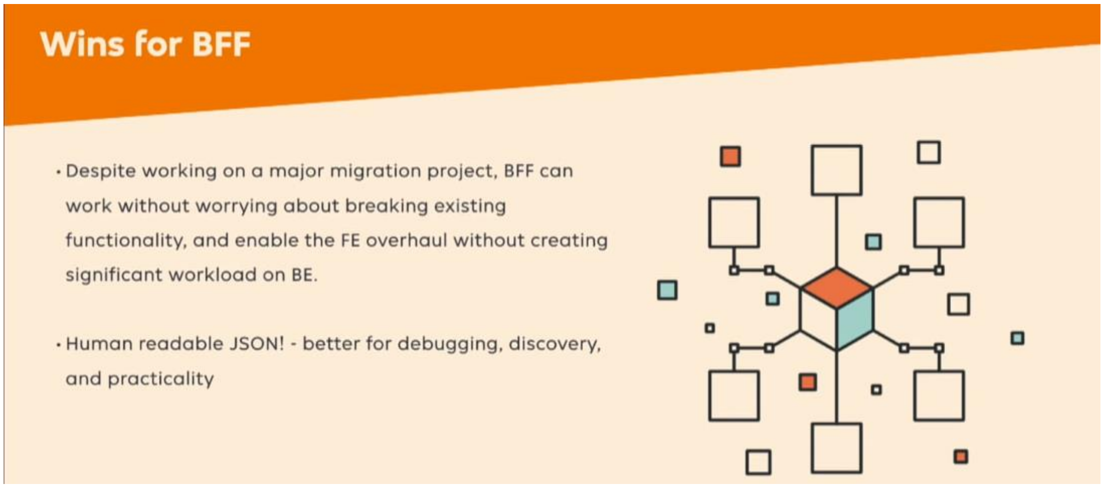
Legacy system





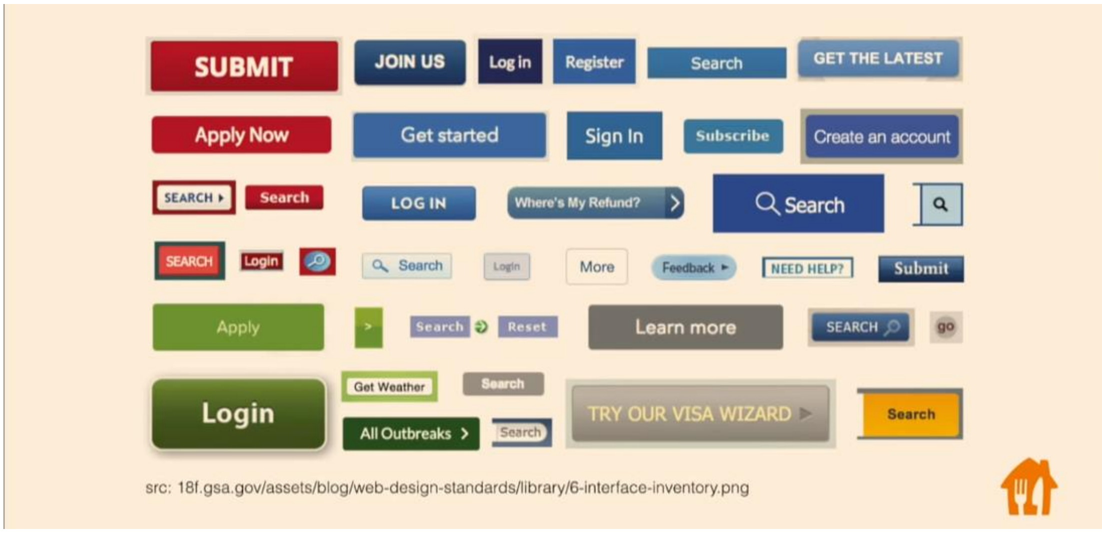
Challenges for BFF

- Having to do status-quo discovery in parallel with anticipating changes in the backend, as we also intend to move towards a service based architecture
- Have to reevaluate and possibly reengineer our dependencies
- To drive API development, we have to accept that we will have to iterate a lot - sometimes meaning rework!



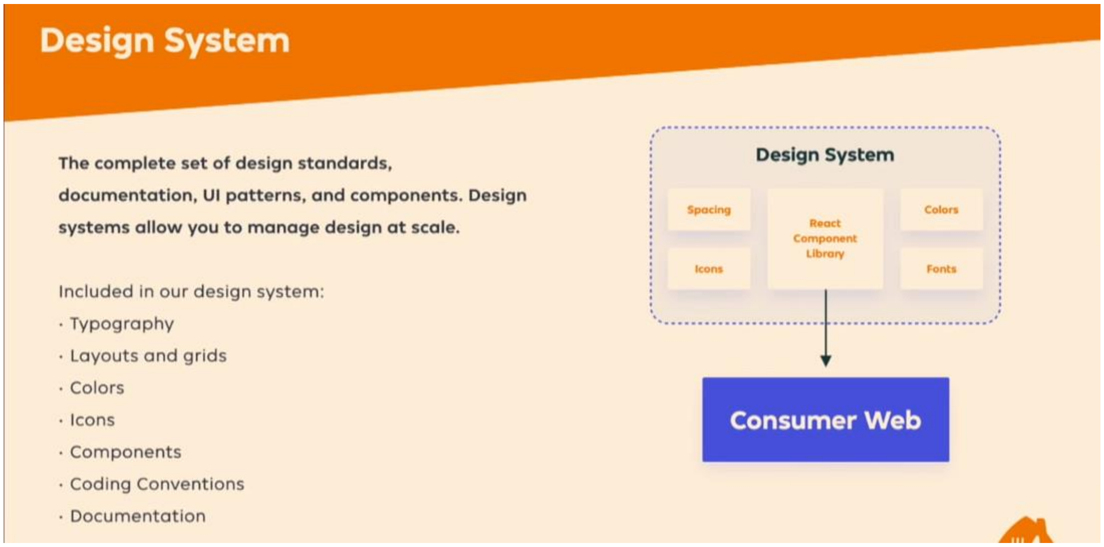
Wins for BFF

- Despite working on a major migration project, BFF can work without worrying about breaking existing functionality, and enable the FE overhaul without creating significant workload on BE.
- Human readable JSON! - better for debugging, discovery, and practicality



src: 18f.gsa.gov/assets/blog/web-design-standards/library/6-interface-inventory.png

This requires a centralized design framework for the company applications for consistency and uniformity.



Design System

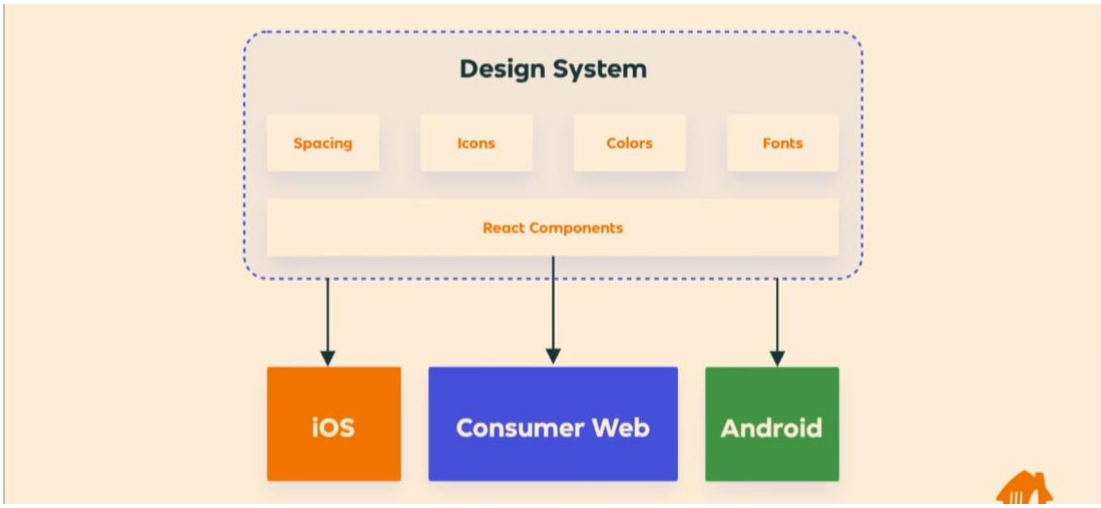
The complete set of design standards, documentation, UI patterns, and components. Design systems allow you to manage design at scale.

Included in our design system:

- Typography
- Layouts and grids
- Colors
- Icons
- Components
- Coding Conventions
- Documentation

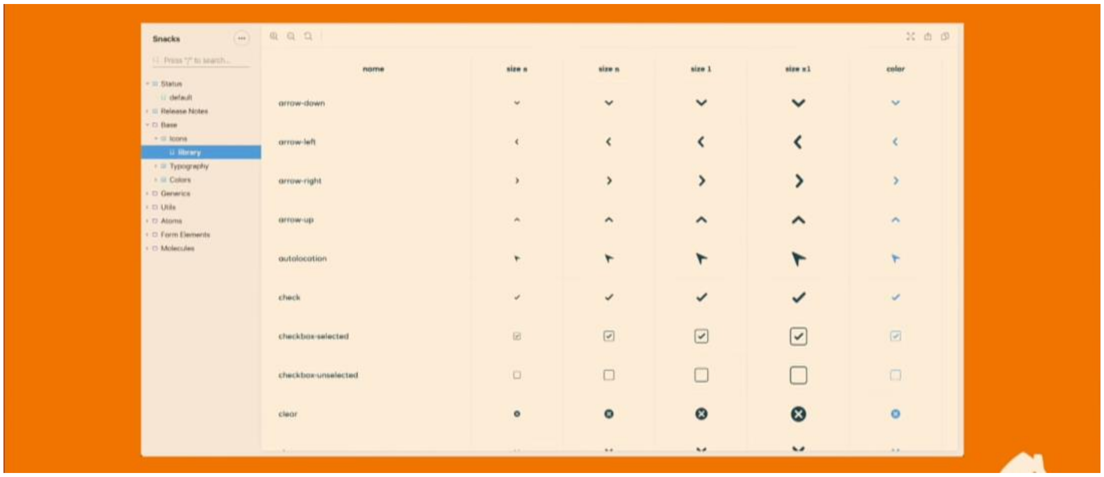
Design System
Spacing
React Component Library
Colors
Icons
Fonts
Consumer Web

The design system team is working on these and the React Component Library



Snacks Design System

We use **Storybook** to house our re-useable React components along with documentation and library for colors, icons, etc. **Frontify** serves as our top-level brand management tool, **Zeplin** is used to host our mockups for the different UIs the teams are working on.



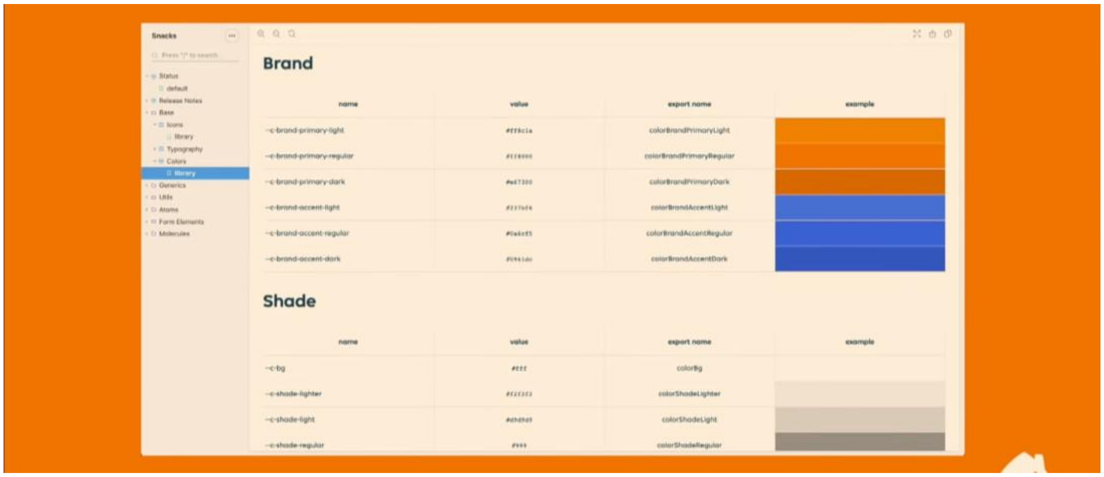
Snacks

Press "/" to search...

- Status
  - default
- Release Notes
- Base
  - Icons
    - library
  - Typography
  - Colors
- Generics
- Utils
- Atoms
- Form Elements
- Molecules

name	size s	size m	size 1	size x1	color
arrow-down	⌄	⌄	⌄	⌄	⌄
arrow-left	‹	‹	‹	‹	‹
arrow-right	›	›	›	›	›
arrow-up	^	^	^	^	^
autolocation	➤	➤	➤	➤	➤
check	✓	✓	✓	✓	✓
checkbox-selected	☑	☑	☑	☑	☑
checkbox-unselected	□	□	□	□	□
clear	⊗	⊗	⊗	⊗	⊗

This is the icon library



Snacks

Press "/" to search

- Status
  - default
- Release Notes
- Base
  - Icons
    - library
  - Typography
  - Colors
    - library
- Generics
- Utils
- Atoms
- Form Elements
- Molecules

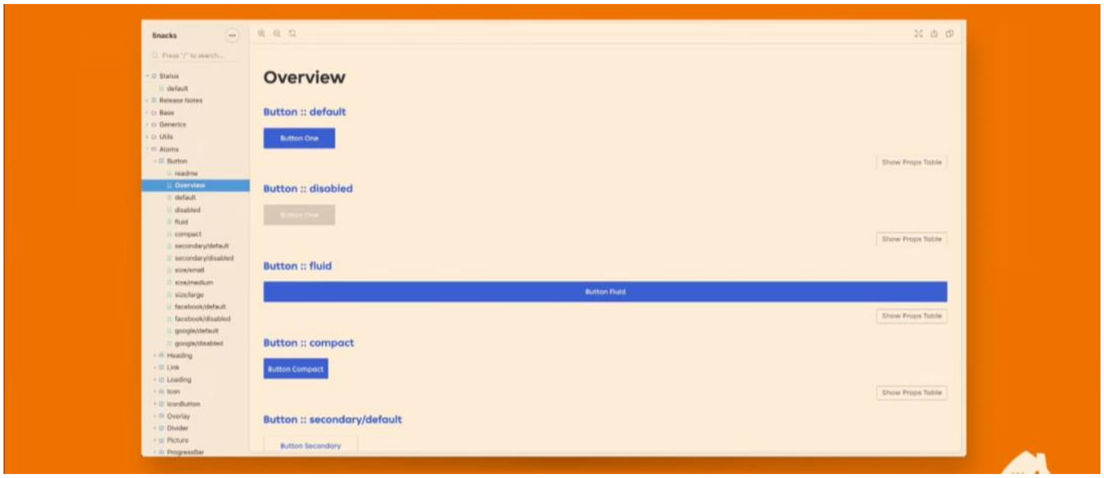
Brand

name	value	export name	example
--c-brand-primary-light	#ff8c1a	colorBrandPrimaryLight	🟧
--c-brand-primary-regular	#ff8000	colorBrandPrimaryRegular	🟧
--c-brand-primary-dark	#e67300	colorBrandPrimaryDark	🟧
--c-brand-accent-light	#3370e6	colorBrandAccentLight	🟦
--c-brand-accent-regular	#0a4ce5	colorBrandAccentRegular	🟦
--c-brand-accent-dark	#0941dc	colorBrandAccentDark	🟦

Shade

name	value	export name	example
--c-bg	#fff	colorBg	
--c-shade-lighter	#f2f2f2	colorShadeLighter	▨
--c-shade-light	#d9d9d9	colorShadeLight	▨
--c-shade-regular	#999	colorShadeRegular	▨

Color library



Snacks

Press "/" to search...

- Status
  - default
- Release Notes
- Base
- Generics
- Utils
- Atoms
  - Button
    - readme
    - Overview
    - default
    - disabled
    - fluid
    - compact
    - secondary/default
    - secondary/disabled
    - size/small
    - size/medium
    - size/large
    - facebook/default
    - facebook/disabled
    - google/default
    - google/disabled
  - Heading
  - Link
  - Loading
  - Icon
  - IconButton
  - Overlay
  - Divider
  - Picture
  - ProgressBar

Overview

Button :: default

Button One

Show Props Table

Button :: disabled

Button One

Show Props Table

Button :: fluid

Button Fluid

Show Props Table

Button :: compact

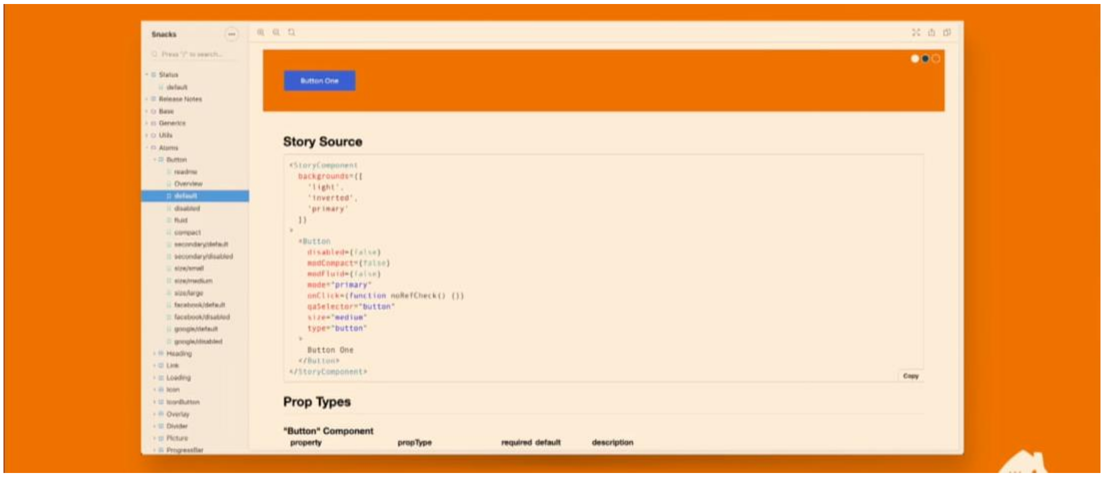
Button Compact

Show Props Table

Button :: secondary/default

Button Secondary

Component library



Story Source

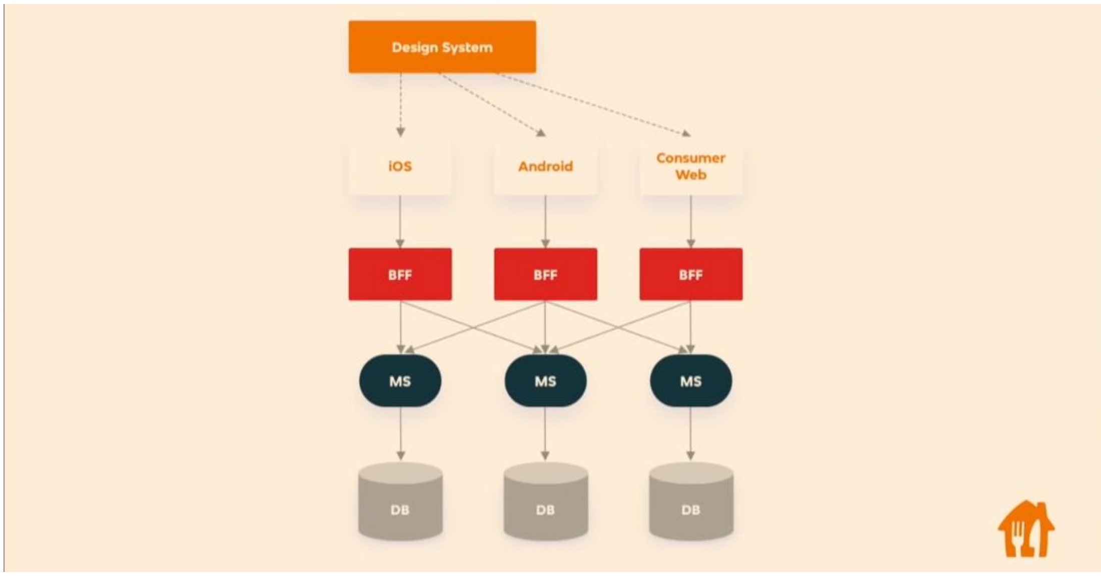
```
<StoryComponent
  backgrounds={[
    'light',
    'inverted',
    'primary'
  ]}
>
  <Button
    disabled={false}
    modCompact={false}
    modFluid={false}
    mode="primary"
    onClick={function noRefCheck() {}}
    qaSelector="button"
    size="medium"
    type="button"
  >
    Button One
  </Button>
</StoryComponent>
```

Prop Types

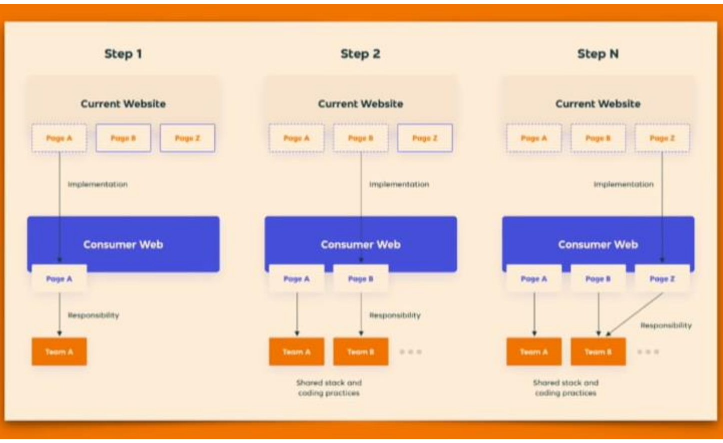
"Button" Component

property	propType	required	default	description

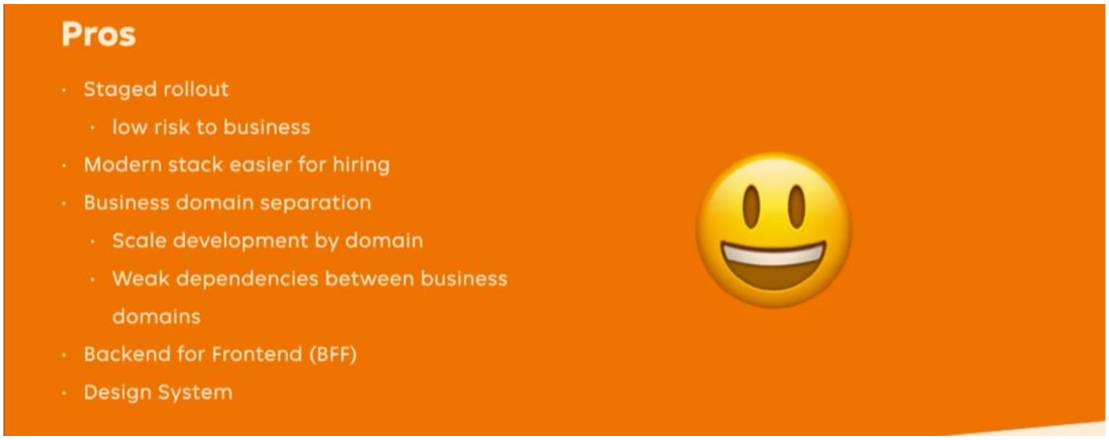
Documentation and implementation details for the React components



Approach



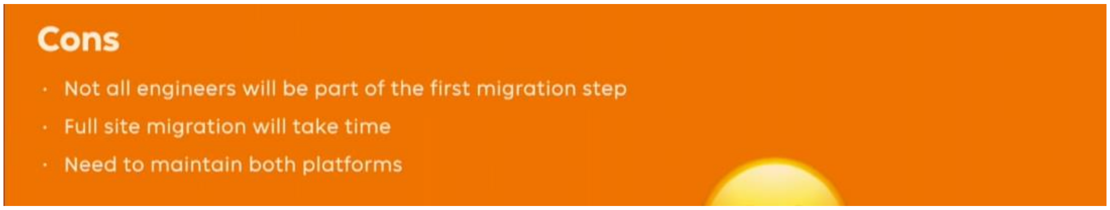
We decided to implement the migration on a page-by-page basis in phases, like rebuilding the menu and checkout pages and processes. We then gradually route users to the new page and remove the old version. This allows us to have a team work on a page at a time.



Pros

- Staged rollout
  - low risk to business
- Modern stack easier for hiring
- Business domain separation
  - Scale development by domain
  - Weak dependencies between business domains
- Backend for Frontend (BFF)
- Design System

😃



Cons

- Not all engineers will be part of the first migration step
- Full site migration will take time
- Need to maintain both platforms