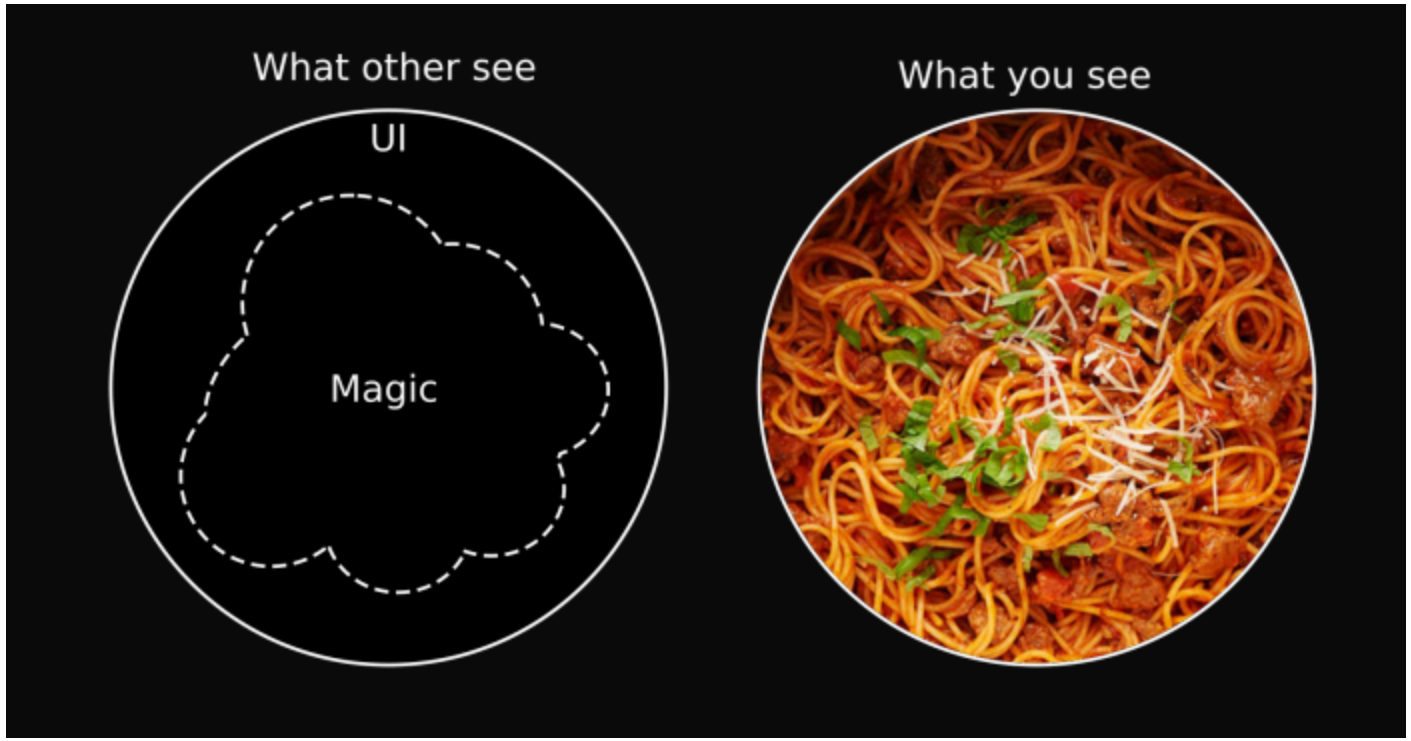
Comment je n'ai PAS refondu mon application ?

Le contexte

- Pogues, c’est une appli front en React… vieille de 2015 🎂 (#POC en prod)
- Redux, un gros router maison, beaucoup de code legacy.
- 

Le contexte

- Elle a grandi vite, sans beaucoup d'encapsulation, avec des dépendances parfois non maintenues.
- Un bon gros monolithe, de la logique métier partout dans le code
- Des devs qui passent et qui se lassent
- **Refonte inévitable ?**


If it works, don't touch it.

Les fausses bonnes idées

- On a eu l’idée, bien sûr : “**et si on refaisait tout avec les dernières technos ?**”
- Sauf qu’une refonte, c’est comme un iceberg. On voit le début, mais on ne voit pas l’étendue.
- On gèle les features. On recrée tout. On débug tout.
- Bref, on prend le risque de livrer une V2… qui n’est jamais meilleure que la V1 et qui n'est jamais livrée...

On ne voulait pas ça.

“On voulait du neuf sans faire table rase.”

L’idée : les micro-frontends

On a opté pour une stratégie plus pragmatique :

- On a testé autre chose : les micro-frontends.
- L’idée est simple : au lieu de tout refaire, on découpe l’IHM.
- Créer une nouvelle app dans le même repo, dans un dossier `next/`
- Qui coexiste avec l’ancienne.
- Et on redirige certaines fonctionnalités vers `next/`, petit à petit.

Concrètement

Pogues est devenu une application hybride :

- L’ancien Pogues continue de tourner,
- Le nouveau qu'on appelle "Next" est monté sur une autre route
- Ils partagent certains composants via une lib interne
- Et on a des points de bascule avec des liens vers next/…
-  Utilisation de module-federation: *Make it easier to share code between javascript applications and make team collaboration more efficient*

Défis techniques de la cohabitation

Évidemment, ce n’est pas magique.

Il a fallu :

- Isoler le CSS pour éviter les effets de bords, (tailwind)
- Gérer deux contextes React différents
- Gestion d'état: tout passe par le modèle commun (Pogues-Model), par le back
- Gérer le routage global

- 🤼 Et côté équipe : **accepter l’imperfection**, travailler dans deux technos à la fois, maintenir un peu plus de code.
- 🚀 Mais ça nous a permis de **livrer en continu**. D’avancer sans bloquer la prod.

Ce qu'on a gagné, le résultat

✅ On a livré des nouveaux écrans modernes, testés, accessibles.

✅ On a gardé les anciennes fonctionnalités disponibles.

✅ L'onboarding des nouveaux dev devient facilité avec la nouvelle stack.

✅ Des devs motivés !

✅ On a évité la dette d’une refonte incomplète.

🚀 **Et surtout : on n’a jamais stoppé la prod !**

Quelques retours utilisateurs

❝ Ça me semble plus clair et intuitif ❞ - *Utilisateur historique*

❝ C'est très séduisant ! ❞ - *Nouvel utilisateur filière*

❝ Bravo ! Belles évolutions ! ❞ - *MOA d'enquête*

❝ Bonjour l'équipe, ..., magnifique la dernière montée de version, ..., vous êtes supers ! ❞

❝ Depuis plusieurs semaines, il y a plein de chouettes petites évolutions ❞

❝ Votre outil Pogues, ... c’est le meilleur du marché et de loin ! ❞ *Consultant européen*

Conclusion

La leçon ?

- 👉 On n’a pas refondu Pogues. On le transforme, en douceur.
- 👉 Et c’est ça qui l’a sauvée.
- 👉 Le micro-frontend, ce n’est pas un dogme. C’est un levier.
- 👉 **C’est imparfait, mais c’est viable, stable, vivant, et provisoire**

Merci de votre attention

Si vous avez une appli legacy, posez-vous la question :

❝ Et si on n’avait pas besoin de tout jeter ? ❞

Des questions ?

Liens et docs:

- Navigating Frontend Migration - Medium
- Module-federation
- Présentation
- *Images libres de droit*