爆速でプロダクトをリリースしようと思ったらマイクロフロントエンドを選んでいた

自己紹介

- 株式会社カケハシ
- ソフトウェアエンジニア
- Nokogiri（@nkgrnkgr）

背景

- 株式会社カケハシとそのプロダクトのMusubiに関して

創業からPMF

- 株式会社カケハシは医療系のSaaSを提供する会社です。主に薬局向けの業務システムを提供しています。
- **Musubi** は創業時のプロダクトです。
- 薬剤師は普段、患者とのやりとりを **薬歴** として残している（お医者さんのカルテのようなもの）。
- Musubi は元々手書きで書いていた **薬歴** をシステム化したことで PMFした。

市場の状況

- Musubi は今では市場の **20%** の薬局で利用されている。
- ただしリリースしてから、後発の競合のプロダクトがリリースされており、以前ほど市場での優位性を失っている。
- 生成AI関連機能では、他社に後れを取っている状態です。
- カケハシとしてはユーザーにとって価値のある生成AIプロダクトを市場に投入して優位性を取り戻したい。

開発の制約

- Musubi はリリースして既に10年近く経っており、プロダクトとして安定はしているが、技術的負債が貯まっていたり、素早く機能開発ができる状態ではなくなっている。
- 特にリリースサイクルは月に1回で、薬局の基幹システムでもあるため障害発生時の影響も大きい。
- ユーザー操作の慣れもあるため、UIを大きく変えることも簡単ではない。
- PharmacyAI(新しい生成AIプロダクト) は実験的な機能であり、素早くユーザーに提供してフィードバックサイクルを高速に回したい。
- MusubiはAngular製のプロダクトで、PharmacyAIチームのメンバーはReactでの開発に慣れている。

最大の課題

- このリリースサイクルの速度差が最大の課題。

マイクロフロントエンドという戦略

- Angular製のアプリケーションの中にReactアプリを同居させる戦略を採用した。
- ユーザーはMusubiと同じプロダクトとしてシームレスに利用できる。
- 開発チームとしては、独立して開発・リリースができることを最優先とする。

そのための戦略

- UI: チャットのような独立したUIにすることで、Musubi側の変更の影響を受けづらく、またPharmacyAI側もMusubiに影響を与えづらい
- 体制: Musubiの開発チームとは別で、PharmacyAIの開発チームを作る
- 技術: Reactに長けたメンバーがReactで開発

帰結

- 結果的にコンウェイの法則に帰結する

想定される疑問

- MusubiのAngularで生成AI機能を作るという選択肢はなかったのか？
  - もちろんあったが、実験的な生成AIプロダクトをリリースサイクルの長いMusubiの中に入れることで、価値の検証が難しくなる。
  - 独立して作るならAngularで作る必要もないと判断

マイクロフロントエンドの具体的な技術

アーキテクチャ

- `vite` でビルドした Reactアプリの `JS` と `CSS` を事前にホスティング
- Angular のアプリにあらかじめ `<div id="react-component" />` を用意し、`JS`、`CSS` ファイルをロードする

iframe も選択肢にあったか？

- ユーザーから見てMusubiとPharmacyAIは一つのアプリ
- 認証機能を別で作る必要はなく、むしろMusubi側に任せたい
- ロードされるJSファイルにバージョン管理も不要だった

独立したUI

- Musubiの画面の上に配置する独立したチャットのような見た目
- Musubi本体からUI的にも独立しているため、お互いに影響を受けづらい

アプリケーション間の通信

- アプリケーション間の通信には `CustomEvent` を利用

CustomEventの利用方法

- 具体的なやりとりの一つをサンプルとして紹介
- Angular（親）とReact（子）のアプリケーション間で相互に通信する方法
- コンテキストの同期。Musubi側でページ遷移があった場合にPharmacyAI側もページ遷移に追従する。（逆のパターンも）
- チーム間のインターフェースをそろえるための型定義
- 認証トークンの有効期限切れチェック、通信時の再認証など

課題

開発・デバッグ環境

- MusubiとPharmacyAIのどちらで不具合が起きているかの切り分けが難しかった。
- CustomEvent のやり取りのデバッグが難しく、イベントの流れを可視化するために**専用のChrome拡張機能を作成した**エピソードを話す。
- Musubiのローカル開発環境を準備しなくても開発ができるように読み込むURLを差し替えるChrome拡張機能を作成した。

実装の泥臭い対応

- PharmacyAIが生成した薬歴をMusubi側に自動挿入する際、Musubi側の画面遷移に追いつけず、**3回までリトライする**という泥臭い実装になった。
- コミュニケーションコストを減らすために、PharmacyAIチームが自らMusubi側のコードを修正し、PRを投げた経験を共有する。

UIの複雑性

- MusubiのグローバルなCSSがPharmacyAIに影響を与えたり、`z-index` の管理が複雑になったりした。

まとめ

- **この選択の成果**:
  - 多くの苦労はあったものの、この選択によって**開発から4か月でリリース**できた。
  - 異なる技術スタックを持つチームが独立して開発を継続できる体制が整った。
- **結論**:
  - このアーキテクチャは、「美しい」アーキテクチャを目指したものではなく、**ビジネスの成長を最大化するための現実的な選択**だった。
  - フロントエンドエンジニアは、ビジネスの要求に応えるために、このような選択肢もあるということを理解してほしいというメッセージで締めくくる。