

# 新規開発と並走した リファクタリング戦略

株式会社 AI-Shift Frontend Engineer

Kazuma.Kurisaki (@KK\_sep\_TT)



Who am I

# 栗崎一真 (Kurisaki Kazuma)

2023.5 AI-Shift 中途入社 フロントエンドエンジニア

- **Carrier:** 2021卒 Rakuten → CyberAgent
- **Role:** Chat bot, Voice bot の管理画面の開発、保守、運用
- **Hobby:** ♠ 🎯 🎲 🎪

X: [@KK\\_sep\\_TT](https://twitter.com/@KK_sep_TT)

GitHub: [@Kult0922](https://github.com/@Kult0922)



# 今日話すこと

1. AI-Shift のプロダクト紹介
2. フロントの泥の紹介
3. リファクタの戦略と歩み
4. やってよかった取り組み、ツールの紹介
5. まとめ

# つくっているもの

AI messenger シリーズ

## AIコールセンター領域



こんな管理画面を作っています

The screenshot displays a complex management interface for AI systems, likely for call center operations. It includes:

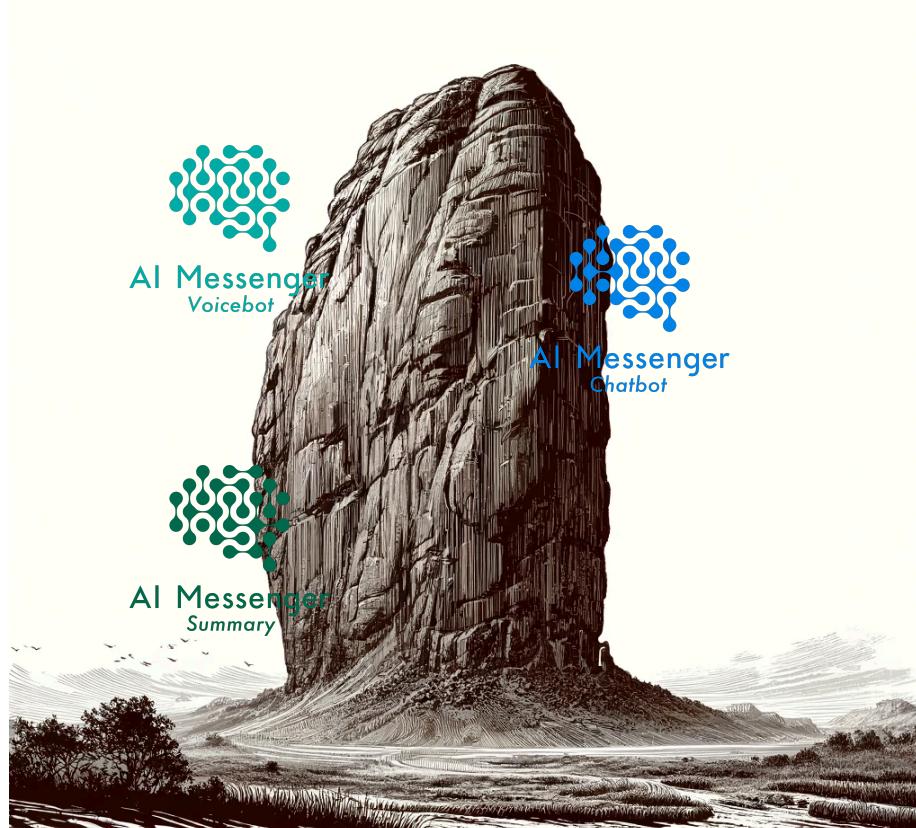
- A main window titled "AI Messenger" showing a list of interactions and a detailed view of one interaction.
- A "問い合わせシナリオ編集" (Inquiry Scenario Editor) window at the bottom, showing a flowchart with various states and transitions.
- A "チャットサポート CX" (Chat Support CX) window on the right, showing a conversation log with a message from a user and a response from the system.

Text elements in the interface include:

- "AI Messenger" (多次に表示)
- "電話番号: 050-5370-2607 H: 23051d39base4a9c0aefb0bf4ff1398002304 2022/06/08 19:06:32"
- "チャット応対を自動化"
- "電話応対を自動化"
- "対話要約を自動化"
- "AI Messenger Chatbot"
- "AI Messenger Voicebot"
- "AI Messenger Summary"
- "問い合わせシナリオ編集"
- "CX X"
- "チャットサポート CX"
- "03-5656-8569"
- "2024/04/10 20:58:59"
- "この中にお聞きになりたいものはあるですか？セカンド"
- "有人チャット接続② (メッセージ修正後呼び出しシーン削除・追加)"
- "電話番号テスト"
- "リンクテキストチェック"
- "メッセージを入力 (Ctrl + Enterで送信)"
- "送信"
- "結構複雑です 😊"

# フロントエンドは一枚岩

Chat bot, Voice bot, Summary のすべてが同じレポジトリで管理されている



# 泥がたまっている

- 🌱 古いバージョンのライブラリ
  - React v16, Node v16, Webpack v4, React Router v5, Fireabasse SDK v7
- 😞 巨大なコンポーネントファイル
  - 1ファイル 3000 行
- 🚧 複雑なアーキテクチャ
  - 関数propsバケツリレーによるロジックのDI
  - カスタム hooks 不使用
  - useEffect Chain
- 📦 複雑な設定のwebpack
  - ミドルウェアの設定
  - 大量の環境変数

# どこから手をつけようか

リスクとリターンのトレードオフ

-  リスク

- 時間をかけても達成できず、時間を浪費する
- 変更の結果、より状況を悪化させてしまう

-  リターン

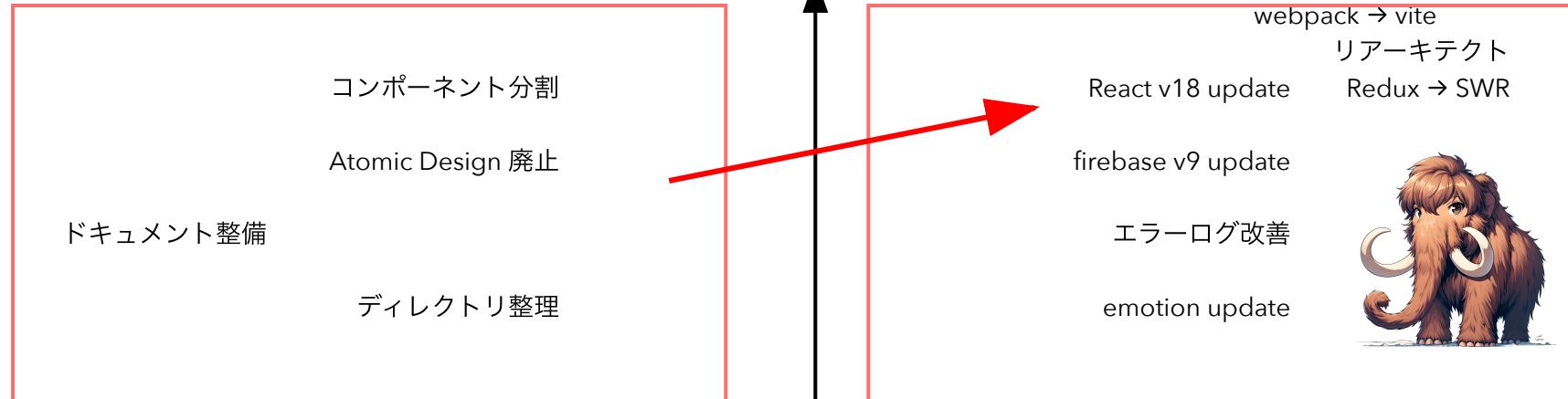
- 今後の開発速度アップ
- コード変更の心理的安全
- 開発がより楽しくなる

おとく！

大物

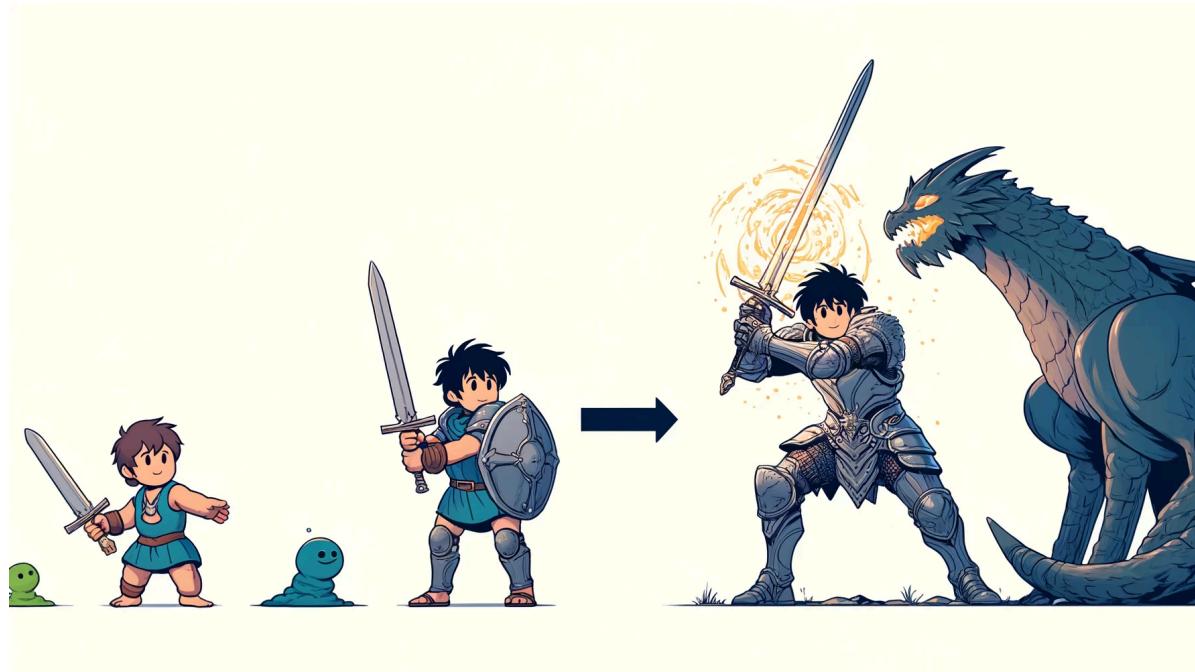
リターン

リスク



# リファクタリングの優先順位

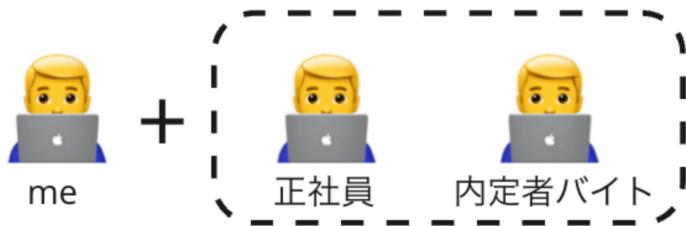
- リターンが大きく、リスクが小さいものから
- ドメイン、コードへの理解が深まってから大物を狩りに行く
- 簡単なリファクタリングでも大きなリファクタリングの土台になる



# 新規開発の波と新メンバー

新規開発が本格的にはじまった

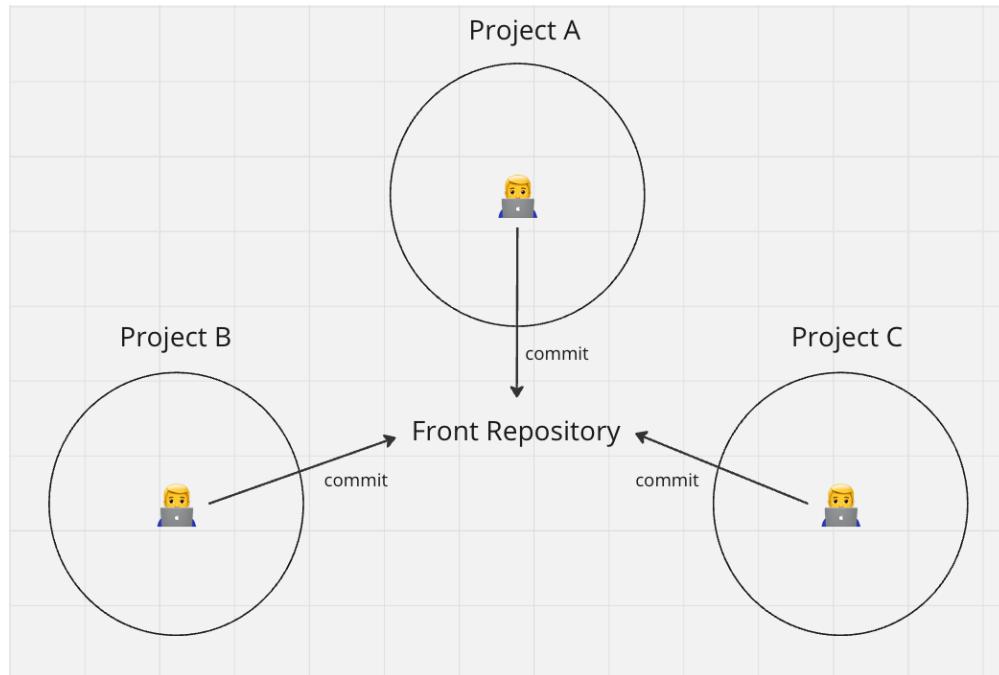
- 同時並行で3つの新プロジェクトが走る
- 新規メンバー加入
  - 正社員2人と内定者バイト1人のチーム



# チームメンバーは別プロジェクトへ

メンバーそれぞれが、異なるプロジェクトに配置

フロントは一枚岩なのでコードベースは同じ



# 新規開発と並走したリファクタリング開始

チームメンバーで定例を週1で開催

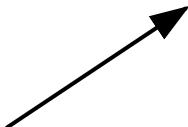
## ■ 話すこと

- 保守、運用の目標を決める
- 新規開発にあたり、優先して改善したいことを共有
- コーディングガイドラインの策定
- 遭遇したコードの問題の共有

## ■ 話さないこと

- 各プロジェクトのタスクの内容
- 各プロジェクトの進捗
- 各プロジェクトの問題

こんな感じのページを作ってました



aim-webとは

- 管理画面とwidgetを開発しているリポジトリです。
- package管理にはyarn workspaceを使っています。
- @aim/console: 管理画面
- @aim/widget: ChatBotのwidget
- @aim/shared: sharedです

運用

PRお作法

refactorとfeatureの切り分け方は以下の観點に従う:

- 最初からリファクタと機能開発のコミットを分ける
- 開発ブランチに向けてリファクタPRを生成する
- 一旦、機能開発とリファクタをした後にコミットを絵面に再実装する
- あきらめて同期的なコミュニケーションをとりながらレビュー
- File Changesにリファクタ範囲をコメントする

QA

master を stage に手動デプロイ → QA開始 → 問題がなければ本番リリース

レビュー

基本1人のapproveがあればマージOK  
ただし変更に不安があるPRは変更周辺ドメインに詳しい人のapproveが必要

OKR

ギャラリービュー

OKR管理

2人でメンテが回るくらいに良い設計にしていきたい

+ 新規

問題調査

解決してなくてもいいので、カジュアルに調査ログとして残していきましょう。

問題の調査ログ

名前 タグ 作成日時

# 新規開発するときも改善を積極的に行う

新規開発で触る部分は可能であれば、リファクタリングする方針に

- 既存のコードの課題が多く、リファクタをした方が開発が早くなるケースも
- リファクタを行うかどうかは、**実装者の判断** に任せて柔軟に対応

# レビューの戦略

1. 最初からリファクタと機能開発のコミットを分ける
2. 一旦、機能開発とリファクタをした後にコミットを綺麗に分ける
3. すべてをあきらめて 同期的なコミュニケーションをとりながらレビュー

レビューの仕方を決めておくことで、安心してリファクタをPRに含めることができた

そして、なにより

リファクタリングをしてくれたメンバーに感謝 🙏

# フロントエンドのリアーキテクト

- ディレクトリ構成
  - Atomic Design → Feature ベース
- サーバーキャッシュ管理
  - Redux → SWR
- ローカル state 管理
  - コンポーネントにベタ書き → カスタムhookに分離
- 関数の多段バケツリレー
  - props から関数を受け取る → カスタムhookを通して関数を受け取る

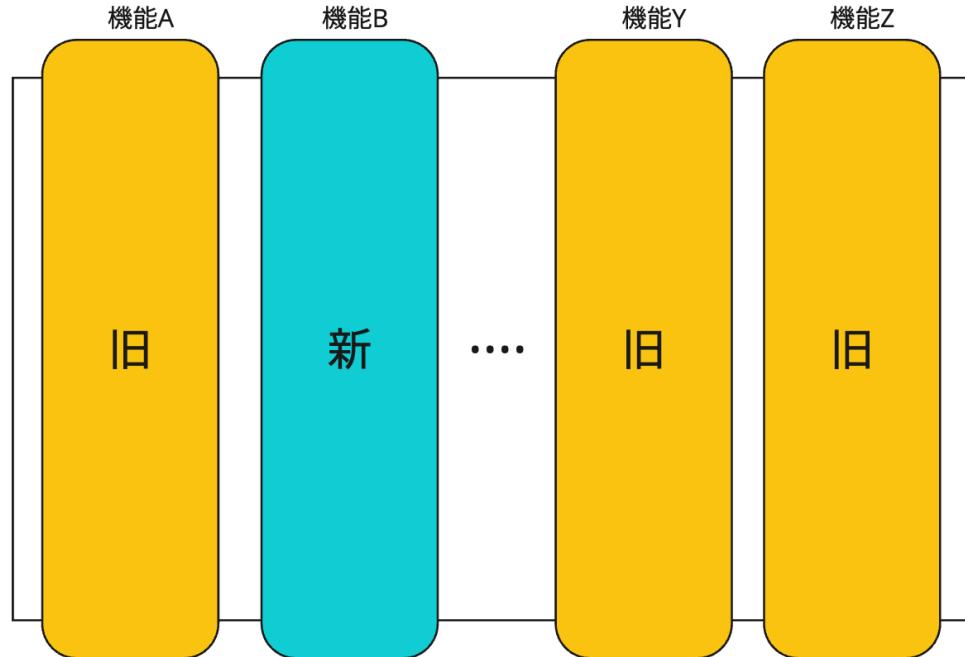
かなりやることがある 😞

すべての機能に対してリアーキテクトをするのは現実的ではない

# フロントエンドのリアーキテクト

一部の機能に対してリアーキテクトを実施

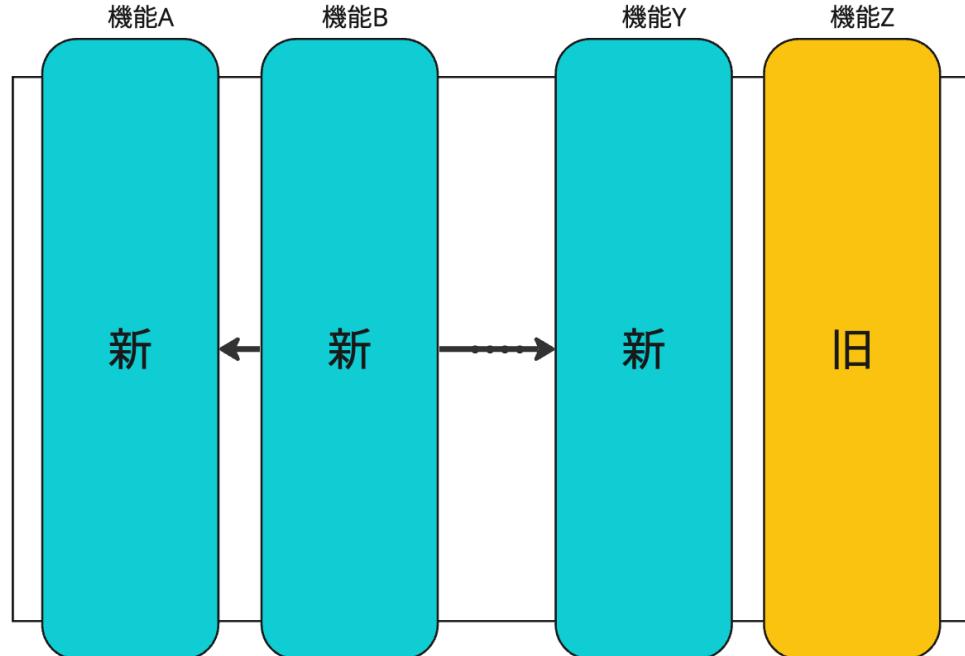
データの取得 → 画面に表示 → ユーザアクション → サーバリクエスト → 画面更新 の一連の流れをリファクタ



# フロントエンドのリアーキテクト

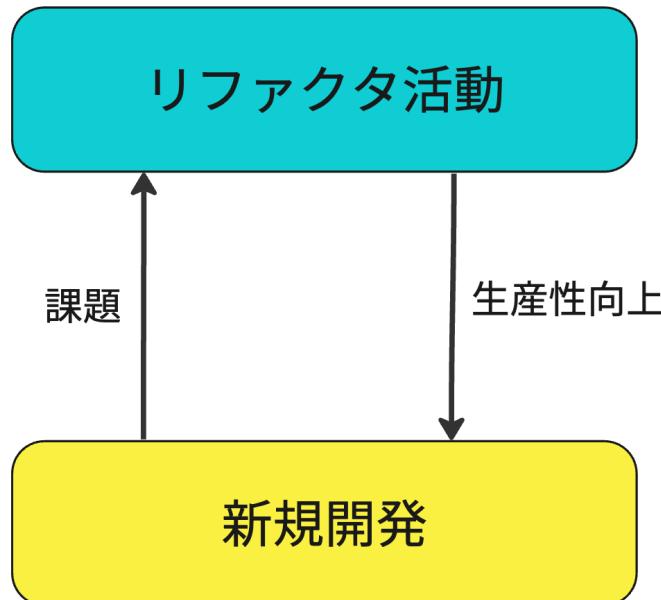
1つの指標ができたらそれに合わせてく

- 既存機能に変更を加えるときに可能ならリアーキテクトする
- 新機能開発のときは新アーキテクチャで実装する



# 新規開発とリファクタのサイクル

- 変更を加えようと開発している時はそのドメインに対して真剣に考えているとき
- 新規開発で発生する新鮮でリアルな課題をリファクタ活動で解消
- リファクタの結果、新規開発の生産性が上がる



# 改善活動を助けてくれたツール

- Notion
  - チームの活動ログ、定例まとめ
- Linear
  - 取り組むタスクの優先度、進捗を管理
- Vite Press
  - コードのドキュメント管理



## タスク管理ツール

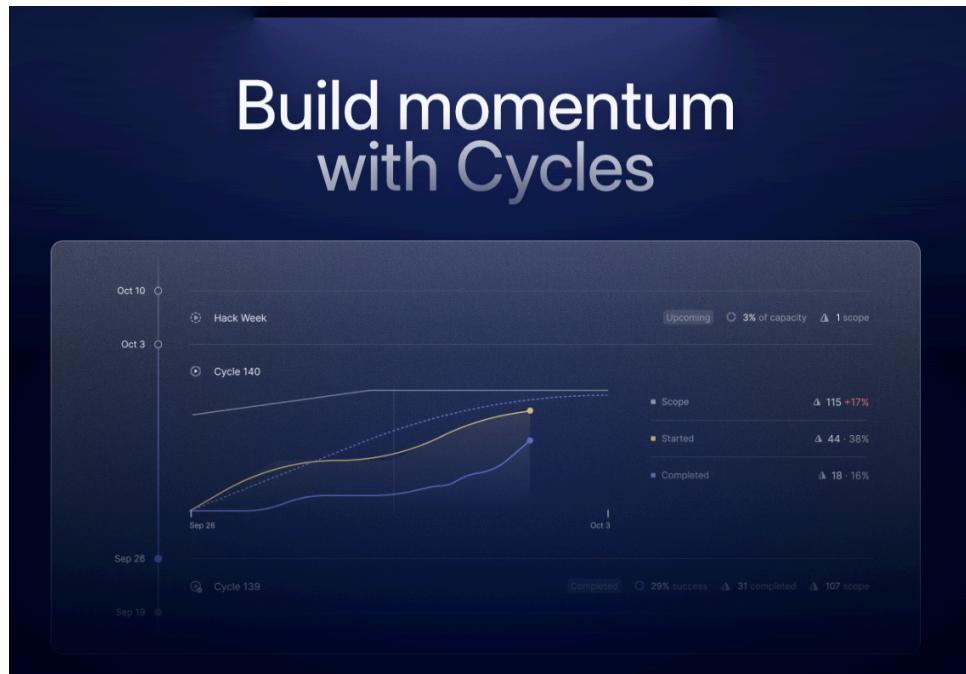
### 特徴

- とにかく高速
- 豊富なショートカット
- ベロシティの可視化

A screenshot of the Linear interface showing a backlog of tasks. The tasks listed are:

- sentry ログ監視の改善
- docs 運用を始める
- console webpack → Vite 移行
- firebase v9 への完全移行
- E2E テストの導入
- コンソール上のwarningを消す
- コミットごとに走る CI を 10 分以内に抑えたいな
- 日付周り、きついでのライブラリの技術選定がした
- npm-run-allを消す

Each task has columns for Title, Roadmaps, Health, Lead, Target Date, and Status (e.g., 9%, 78%, 25%, 0%). A pink banner at the bottom reads "Project に改善タスクを積んで優先度、担当者を管理" and "APIキャッシュのstate管理をSWRにしたい".



# VitePress



- 設計やコーディングガイドラインをドキュメント管理
- いい感じの見た目のドキュメントが簡単につくれる
- コードと一緒にGit管理

aim-web architecture

アーキテクチャ・コード解説のためのドキュメントです。

Getting start

Products

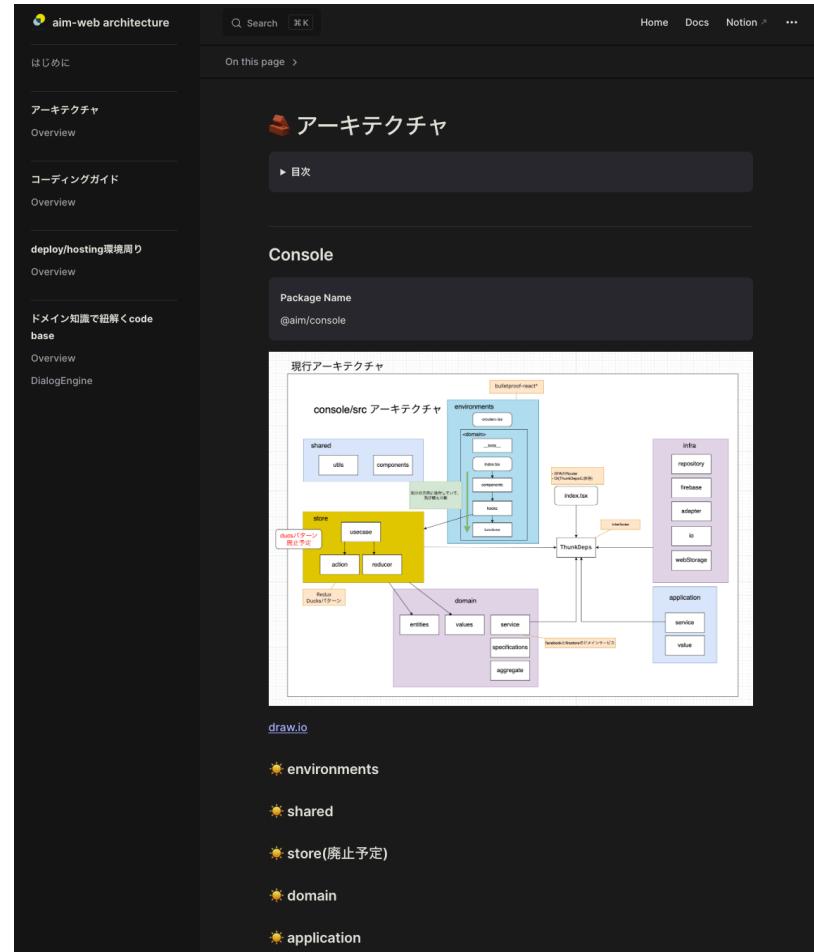
VoiceBot, ChatBot, Summary の管理画面のフロントエンド画面、およびSDKを開発しています。

Team

Kazuma Kurisaki  
Creator

Junki Yoshida  
Creator

Hibiki Mizuno  
Creator



# まとめ

新規開発と並走したリファクタリング戦略

- リファクタリングは ローリスク、ハイリターンなもののから 取り組んだ
- ターゲットを絞って リアーキテクトを実施
- 機能変更、新規機能開発のときに可能なら新アーキテクチャで実装
- リファクタ込みのコード変更のレビューの方針 を決めた

DXを改善しながら機能開発をするのは楽しい！

ご清聴ありがとうございました！

