職務経歴書

分量の関係で一部省略、完全版は下記に記載。
https://github.com/KazukiHayase

基本情報

項目	詳細
氏名	早瀬和輝
生年月日	1999/01/02
居住地	東京都
GitHub	https://github.com/KazukiHayase
X	https://x.com/KazukiHayase
Zenn	https://zenn.dev/kazu777
Speaker Deck	https://speakerdeck.com/Kazukihayase

職務要約

これまで株式会社BuySell Technologiesにて、約4年間にわたりフルスタックエンジニアとして、Google Cloud、Go、GraphQL、React を中心に、インフラ・バックエンド・フロントエンドの開発に従事。

リユース特化のOMS開発では、プロジェクトマネージャー兼テックリードとして技術選定や開発プロセスの整備を推進し、開発効率向上に貢献。PdMと協力し、ユーザー視点を重視した開発を実践。

今後は、技術戦略やプロダクト戦略にも関与し、よりレイヤーの高い意思決定に携わりたいと考えている。

技術スタック

項目	詳細
インフラ	Google Cloud、AWS、Terraform、SAM
データベース	PostgreSQL、DynamoDB
バックエンド	GraphQL、Hasura、OpenAPI、OpenAPI Generator、oapi-codegen、Go、Ruby on Rails、Elasticsearch
フロントエンド	React、Next.js、TypeScript、Apollo Client、GraphQL Code Generator、React Hook Form、zod、MUI、Storybook
CI・CD	GitHub Actions、CircleCI
ツール	Jira、Confluence、Slack、Sentry

自己PR

ユーザー志向での開発

「なぜやるか」を常に考え、ユーザーに価値を提供することを重視して開発を行っている。 技術が好きでエンジニアリングに携わり続けたいという思いはあるが、それ以上にユーザーの課題を解決したいという気持ちが強い。 実際に、開発業務を一時的に離れ、ユーザーインタビューや要件定義などのPdM業務に専念した期間もある。 プロダクトを推進するために必要なことは、役割にとらわれず何でも実行してきた。

リード力

チームの課題を発見し、解決するリード力を持っている。 リーダーやプロジェクトマネージャーとして、チーム全体を俯瞰し、課題を見つけ解決することに取り組んできた。 チームが抱えている優先度の高い課題に対しては、メンバーを巻き込みながら、チーム全体で解決策を考え、実行することを主導してきた。

また、チームに関する意思決定を数多く行ってきた。 意思決定の際には、目的を明確にし、判断軸とトレードオフを整理することを心掛けている。

アウトプット力

仕様理解力の高さ、キャッチアップ力、概念整理の速さを強みとし、個人としてのアウトプット量も多い。 プロジェクトマネジメントや採用といった開発以外の業務も行いながら、以下のような成果を維持している。

期間	PR作成数	レビューしたPR数
2024年	806 PR	3993 PR
2023年	1028 PR	3276 PR

今後やりたいこと

レイヤーの高い意思決定を行う

事業やプロダクトの方向性を決めるような、大きな意思決定を行えるようになりたいと考えている。 視座を上げる必要があると感じており、機会があれば技術戦略やプロダクト戦略に関わる業務にも携わりたい。 そのため、規模の大きい会社よりも、ワンプロダクトの会社や規模が小さめな会社で、責任を持ってプロダクトの成長に深く関与できる環境を希望している。

技術を深める

より上位のレイヤーで意思決定を行うためには、技術力と引き出しの多さが必要であると考えている。 現時点では、技術の幅や深さが十分ではないと感じているため、今とは異なるドメイン領域での開発や、高難易度の課題解決に取り組むことで技術力を高めていきたい。

職務経歴

2021/04 ~ 現在 株式会社 BuySell Technologies

リユース特化のOMS（Order Management System）の開発

概要

項目	詳細
期間	2021/07~現在
チーム構成	EM 1名、PdM 1名、バックエンド 8名、フロントエンド 2名
役割	プロジェクトマネージャー、テックリード
使用技術	**インフラ** Google Cloud、Terraform、PostgreSQL **バックエンド** GraphQL、Hasura、OpenAPI、oapi-codegen、Go、gqlgen、GORM、Elasticsearch **フロントエンド** React、Next.js、TypeScript、Apollo Client、GraphQL Code Generator、React Hook Form、zod、MUI、Storybook

プロジェクト内容

複数のECサイトへの出品・受注業務を一元管理するWebアプリケーションの開発。 複数のECサイトへの出品や受注管理を他社SaaSで行っていたが、パフォーマンスが低さや不具合の多さから、事業のスケールを阻害していたため、内製で開発を行うことになった。 自社だけの利用ではなく、将来的にSaaSとして外部に提供していくことも考慮して設計・実装を行った。

主要実績

プロダクト全体の技術選定

バックエンド・フロントエンド両方の技術選定に携わった。

3年間に及ぶ新規プロダクト開発の技術選定の振り返り
もし今からGraphQLを採用するなら

開発生産性向上のための取り組み

チームリーダーとして、スクラムの導入や開発生産性向上のための取り組みを主導した。 具体的には、KPIとしてPR作成数を設定し、KPIを達成するための施策をスプリントごとに検討・実施を行った。

生産性が上がり続けるチームを作るための第一歩
自分だけが頑張るのをやめて、フルスタックなチームを作る

可変項目のDB設計

ユーザーが自由に商品に対して項目や選択肢を定義でき、それを使った複雑かつ高速な検索が求められていた。 この要件を満たすため、商品の可変属性データはPostgreSQLでEAV（Entity-Attribute-Value）を用いて管理し、そのデータをElasticsearchに同期して検索を行う方式を採用した。

バックエンドの自動テストの整備

開発が進むにつれて、モックやテストデータのセットアップが複雑であることや、記法が統一されていないことなど、バックエンドの自動テストに関連する複数の課題が顕著になり、それにより開発効率が低下していた。 そこで、バックエンドの自動テストの整備を行い、開発効率の向上を図った。

Goでテストをしやすくするためにやったこと
GoのTestifyを使って独立したサブテストを実現する
Goのジェネリクスを使って、テーブル駆動テスト(TDT)に統一性を持たせる

フロントエンド構成の整備

開発効率の向上や属人化の防止を目的として、フロントエンドの構成を整備した。 ルールを設けたり明文化することは重要だが、ルールの認知漏れや認識齟齬がどうしても発生してしまう。 そのため、自動生成ツールを整備やCIへの組み込みなど、可能な限り仕組み化・自動化を意識して整備を行った。

開発効率を上げるためのモダンなフロントエンド構成
MUIをベースにしたデザインシステムの構築
graphql-eslintを使用してGraphQLの命名規則を強制するカスタムルールを作る

EC出品用商品撮影アプリのフルリプレイス

概要

項目	詳細
期間	2021/04~2024/08
チーム構成	PdM 1名、iOS 3名、バックエンド 2名
役割	フルスタックエンジニア
使用技術	**インフラ** AWS、AWS SAM、Lambda、DynamoDB **バックエンド** OpenAPI、OpenAPI Generator、Go **iOS** Swift、RxSwift

プロジェクト内容

リユース事業の出品業務における商品撮影アプリのフルリプレイスプロジェクト。

撮影が完了したものから順次出品していくという業務フローであったため、撮影効率が出品効率並びに売り上げに影響していた。そのため撮影効率向上のために業務フロー自体の見直しを行うことになったが、既存アプリが特定の業務フローのみにしか対応していない仕様であったので既存アプリのままでは業務フローの変更が行えない状況であった。 また既存アプリではパフォーマンス・保守性がともに低く、将来的なことも考えた結果、リプレイスするに至った。

3ヶ月で開発~テストまで行い、リリースに至った。 既存アプリからのリプレイスを完遂し、業務時間を20%削減することに成功した。

主要実績

AWS Step Functionsを用いたバッチ処理の実装

AWS Step Functionsを用いて画像の生データを別のS3にコピーし、出品管理サービスのFTPサーバーへアップロードするバッチ処理を実装した。 Lambda単体の責務は小さくし、実行順序はStep Functionsで管理することで、Lambda同士を疎結合にし保守性が高くなるように設計した。

この構成にしたことで、リリース後に連携先を増やすことがあったが、その際も既存のLambdaに手を加えることなく、新規Lambdaの実装とStep Functionsの設定のみで対応できた。