# Resume(details)

## Table Of Contents



## 自己紹介

インターネットでは「かなで」という名前で活動しています。

2018年4月よりWebのエンジニアとしてキャリアを開始しています。

主たる領域はサーバサイド開発とクラウドを使ったインフラ構築、多少はWebフロントエンド開発の心得があります。

TypeScriptやGolangとクラウド、Terraformでのサーバサイド・インフラ開発が得意です。

twitter (https://twitter.com/py_kanade0404)

GitHub (https://github.com/kanade0404)

はてなブログ (https://kana-kanade.hatenablog.com/)

Obsidian Publish (https://publish.obsidian.md/kanade0404/kanade0404+memo)

LAPRAS (https://lapras.com/public/XNYCJEI)

Scrapbox（ほぼ停止） (https://scrapbox.io/pykanade/)

# 経歴一覧

技術スタックは私が関わった範囲のみ記載しており、そのシステムが内包している技術全てではない可能性があります。

## 株式会社Speee (https://speee.jp/)(2022年8月~)

### 広告配信プラットフォームの開発

以下のコンポーネントの設計・開発・運用を行なっていました。

- 広告配信サーバ
- バッチシステム
- 管理画面
- アドタグ
- TerraformによるAWSインフラ

TerraformやNode.jsのバージョンアップ、EC2で動かしていたJenkinsの2.xへのアップグレード、他社との連携機能の要件・設計・実装や新しいアドタグの開発をしています。

プロパーのエンジニアが少ないので、基本的に一人一プロジェクトで設計と実装を全て行う立ち回りが必要になるので、幅広い要素技術を一通り扱えることと個人PMとしてプロジェクト管理をすることが求められます。

<details>

<summary>技術スタック</summary>

- Backend

    - Go (https://go.dev/)
    - Python3.9 (https://www.python.org/)
    - unittest (https://docs.python.org/ja/3/library/unittest.html)
    - pandas (https://pandas.pydata.org/)
    - MySQL (https://www.mysql.com/)
    - Ruby on Rails (https://rubyonrails.org/)
- Infrastructure
    - AWS (https://aws.amazon.com/jp/)
        - EC2 (https://aws.amazon.com/jp/ec2/)
        - Athena (https://aws.amazon.com/jp/athena/)
        - S3 (https://aws.amazon.com/jp/s3/)
        - Sagemaker (https://aws.amazon.com/jp/sagemaker/)
        - ECS (https://aws.amazon.com/jp/ecs/)
- DevOps
    - Terraform (https://www.terraform.io/)
    - GitHub Actions (https://github.co.jp/features/actions)
    - Airflow (https://airflow.apache.org/)
    - Jenkins (https://www.jenkins.io/)

</details>

**SaaS**

開発中のため詳細はリリース後に公開。

以下が対応したスコープです。

- PdMとのユーザーストーリーやリーンキャンバスのブラッシュアップ
- 技術選定
- SaaSシステム設計
- 社内管理画面の要件定義
- API設計・実装
- AWSインフラ設計・構築
- 業務委託エンジニアのタスク・進捗管理
- その他開発全般の意思決定

<details>
<summary>技術スタック</summary>

- App
    - Node.js (https://nodejs.org)
    - TypeScript (https://www.typescriptlang.org/)
    - Hono (https://hono.dev/)
    - React (https://reactjs.org/)
    - Next.js (https://nextjs.org/)
- Infrastructure
    - AWS (https://aws.amazon.com/jp/)
        - ECS (https://aws.amazon.com/jp/ecs/)
        - Fargate (https://aws.amazon.com/jp/fargate/)
        - Aurora (https://aws.amazon.com/jp/rds/aurora/)
        - Elastic Load Balancing (https://aws.amazon.com/jp/elasticloadbalancing/)
        - Lambda (https://aws.amazon.com/jp/lambda/)
        - DynamoDB (https://aws.amazon.com/jp/dynamodb/)

        - Cognito (https://aws.amazon.com/jp/cognito/)
    - Akamai (https://www.akamai.com/)
- DevOps
    - Testcontainers (https://testcontainers.com/)
    - Docker (https://www.docker.com/)
    - Sentry (https://sentry.io/)
    - Datadog (https://www.datadoghq.com/)
- AI Tools
    - Claude Code (https://www.anthropic.com/news/claude-code)
    - Cursor (https://www.cursor.com/)
    - Devin (https://www.cognition.ai/devin)
    - Coderabbit (https://www.coderabbit.ai/)

</details>

## 株式会社良品計画 (https://www.ryohin-keikaku.jp/)(2022年8月~10月)

**ECサイトのバックエンド・フロントエンド開発**

主に2つの業務内容があります。

1つ目はElasticSearchのパフォーマンス改善の調査とパフォーマンス改善をするための修正実装のプロトタイピングを行いました。

2つ目はWebフロントエンドのTypeScript化とStorybookの導入、カタログ作成をしました。

<details>

<summary>技術スタック</summary>

- Backend
    - Java8 (https://www.java.com/)
    - Spring Boot (https://spring.io/projects/spring-boot)
    - PHP (https://www.php.net/)
    - Lumen (https://lumen.laravel.com/docs/9.x)
    - Go (https://go.dev/)
    - MySQL (https://www.mysql.com/)
- Frontend
    - JavaScript (https://developer.mozilla.org/en-US/docs/Web/JavaScript)
    - TypeScript (https://www.typescriptlang.org/)
    - React (https://reactjs.org/)
    - Redux-Saga (https://redux-saga.js.org/)
    - Ant Design (https://ant.design/)
    - Storybook (https://storybook.js.org/)
    - Jest (https://jestjs.io/ja/)

</details>

## 株式会社クラフトマンソフトウェア (https://craftsman-software.com/)
## （2022年5月〜7月）

こちらは業務委託で参画させていただいております。

**大手メーカー様の内製Androidアプリのビルドパイプライン刷新**

業務内容

ビルドパイプラインのインフラで使用しているAWSの構築を担当しました。

<details>

<summary>技術スタック</summary>

- Infrastructure
  - AWS (https://aws.amazon.com/jp/)
    - ECR (https://aws.amazon.com/jp/ecr/)
    - S3 (https://aws.amazon.com/jp/s3/)
- DevOps
  - Terraform (https://www.terraform.io/)

</details>

## マキヤマブラザーズ株式会社 (https://www.makiyamabrothers.jp/)（2022年5月〜7月）

こちらは業務委託で参画させていただいております。

**自社Webアプリケーションの開発**

マキヤマブラザーズ株式会社様が提供している複業サービス「DeLMO」の開発に携わりました。

<details>

<summary>技術スタック</summary>

- Backend
  - Go (https://go.dev/)
  - MySQL (https://www.mysql.com/)
  - gqlgen (https://github.com/99designs/gqlgen)
  - SQLBoiler (https://github.com/volatiletech/sqlboiler)
- Frontend
  - TypeScript (https://www.typescriptlang.org/)
  - Next.js (https://nextjs.org/)
  - Ant Design (https://ant.design/)
  - Apollo Client (https://www.apollographql.com/)

</details>

## 株式会社ビットエー (https://bita.jp/)（2019年8月〜2022年5月）

ビットエーではサーバサイドからインフラの設計・運用、フロントエンド開発の開発業務に携わりました。

SESであることから実際にお客様のエンジニアと会話したり非技術職の方ともコミュニケーションを取って業務を進めておりました。入社当時は課題感がありましたが、現在は指摘されないので非技術職の方とのコミュニケーションは問題ないと思っております。

また開発業務以外では、主に知見共有と採用業務を行いました。

社内で技術知見を広めるために勉強会の主催やLT会での多数登壇をしました。
勉強会は普段業務では直接関わらないが現代の基礎となっている技術について取り上げています。現在はマスタリングTCP/IPを題材にネットワークについて勉強をしています。

採用業務では実際に社員・業務委託の面接に複数回出席して応募者様のスキルをヒアリングしたり会社とマッチングするかを会話しました。
採用面接以外でも毎日良さそうなエンジニアの方を見つけてはGitHubの実装を見て実際に声をかけたりプログラミングテストの採点を行なっておりました。

開発業務のみならず、会社が楽になるように日々動いています。

### 広告配信の最適化の基盤システムの開発・運用

#### 案件クライアント

美容系専門の広告代理店。

#### 期間

2021年1月~2021年12月。

#### 開発人数

2〜5人（途中で増加）

開発全体としてはスクラム開発の手法を一部採用してGit-flowで開発をしました。
1週間を1スプリントとして、スプリントの最初にSprint Planningをしてスプリント最後にRetrospectiveをしてKPTについて話し合いました。
新たに発生したタスクに対してはstory pointをplanning pokerで見積もりをしました。

当初は自分と弊社の別メンバーのみで開発をしましたが、メンターをしていた常駐先クライアントの新卒3名を開発に加えて最終的には5名で開発をしました。

主に2つの開発をしました。

1つは常駐先クライアント様の社内で広告運用に使用している社内Webアプリケーションの機械学習を使った予算配分の最適化機能の開発です。

もう1つは自社顧客向けの効果測定と広告配信の最適化の基盤システム開発です。

社内Webアプリケーションは自分がアサインされる以前に弊社の別メンバーが開発していたアプリケーションにエンハンスで新規のページから作成をしました。

構成はBackendがTypeScript+Expressで、FrontendがTypeScript+Next.jsでCloud Runにデプロイしています。

初めて実務でTypeScriptやExpress、Next.jsやGCPを使いましたが、アプリケーションの仕様を含めてキャッチアップを行い、スケジュール通りに新機能をリリースできました。

基盤システム開発は自社顧客向けにFacebook Conversion APIを利用した広告配信の最適化と効果測定を目的としています。設計から開発運用を行いました。最終的に既存顧客の5社が導入、通販DXを目的としたデータ収集でも10社に導入していただきました。

立ち上げ時は自分含め2名で開発をしました。プロパーではなかったですが、システム要件の策定からクラウドアーキテクチャの設計、開発と運用まで含めて任せていただきました。

現在ではリリース済みで、いくつかの企業様に導入していただいております。

この開発では、多くのGCPサービスを使い倒す機会を得ることができたこと、実務未経験でしたがIaCにTerraformを自分から導入し、インフラを安全に管理できるようにしました。

この2つの開発以外の業務では、広告運用チームとそのチームの担当顧客様とのFacebook Conversion APIの導入相談を受けたり、常駐先クライアント様の新卒のメンターを行いました。

開発面ではIaCの旗振り役をやったりテストコードの推奨、アプリケーションやTerraformのCI/CD自動化を導入しておりました。

<details>

<summary>技術スタック</summary>

- Backend
  - TypeScript (https://www.typescriptlang.org/)
  - Node.js (https://nodejs.org)
  - Express (https://expressjs.com/)
  - Jest (https://jestjs.io/ja/)
  - Puppeteer (https://github.com/puppeteer/puppeteer)
- Frontend
  - TypeScript (https://www.typescriptlang.org/)
  - Next.js (https://nextjs.org/)
  - React (https://reactjs.org/)
  - Jest (https://jestjs.io/ja/)
  - Ant Design (https://ant.design/)
  - styled-components (https://styled-components.com/)
- Infrastructure
  - Google Cloud Platform (https://console.cloud.google.com/)
    - Cloud Run (https://cloud.google.com/run)
    - App Engine (https://cloud.google.com/appengine)
    - BigQuery (https://cloud.google.com/bigquery)
    - Cloud SQL (https://cloud.google.com/sql)
    - Cloud Pub/Sub (https://cloud.google.com/pubsub)
    - Cloud Tasks (https://cloud.google.com/tasks)
    - Dataflow (https://cloud.google.com/dataflow)
    - DataStore (https://cloud.google.com/datastore)
    - Cloud Storage (https://cloud.google.com/storage)
  - MySQL (https://www.mysql.com/)
- DevOps
  - Terraform (https://www.terraform.io/)
  - GitHub Actions (https://github.co.jp/features/actions)

</details>

**AI/VRを活用した不動産仲介プラットフォーム開発**

**案件クライアント**

某外資系コンサルティングファーム。

**期間**

2019年10月〜11月。
2020年4月〜12月。

途中参画にはなりますが、入社して初めての案件でした。

Laravelを使ったバックエンド開発とVue.jsを使ったフロントエンド開発、詳細設計とテストを行いました。

LaravelとAzureは初めてだったためキャッチアップを行い、既存メンバーよりも詳しくなりました。

また以下のような実績があります。

- テストコードの導入
- Clean Architecture likeな設計導入
- Bus EventをVuexに置き換え
- Atomic Designの導入

まずバックエンドのフロントエンドともにテストコードがなかったため、テストコードの追加とテストコードを書く意識を作りました。テストコードを書くことで実装の理解が早まり、CircleCIやAzure Pipelinesとの相乗効果からリグレッションを未然に検知できる仕組みを整備しました。結果として在籍している間はPoCもしましたが障害ゼロを達成できました。

そして設計指針を導入して実践しました。
当初はLaravelのModelとControllerそれぞれにビジネスロジックが混在しており、Fat ControllerとFat Modelが入り混じった状態でした。
永続化層としてRepository層を追加しました。ビジネスロジックはService層に集約しました。外部とのやり取りはInfrastructure層に集約しました。Controller層はHTTPリクエストを受け付けてService層に受け渡し、Service層から受けた結果をHTTPレスポンスとして返すのみとしました。

設計指針を作ったことで全体の設計に一貫性が保たれ、自然とモジュールの責務について考えることができるようになりました。指針もお互いシステム設計に詳しいわけではないので、深入りしすぎることはせず、共通理解として作成して本来のプロダクト開発に専念しました。

フロントエンドでは2つあります。まずBus EventをVuexに置き換えました。

当時はBus Eventでコンポーネントの状態の受け渡しをしていました。これをVuexに置き換えましたが、本来Vuexで扱うべきではない状態も入り込んでしまいました。ある程度は成功ですが、一部は理想とは程遠い状態となってしまいました。あらかじめ現在のstateを洗い出した上で、Vuexに持たせるべきstateは何か、他はコンポーネントの依存関係を整理して解決できないかなどを考えるべきでした。

またフロントエンドも設計指針がなかったのでAtomic Designを導入しました。これも一部では設計指針として分かりやすかったのでよかったですが、安易が導入は微妙だったという結論に至りました。Atomic Designは本来UIデザイナー向けの設計手法であり、エンジニアが導入するなら当時導入していたFigmaのコンポーネントからAtomic Designを踏まえて設計するべきでした。当時はAtomic Designに準拠したデザインではなかったため、atomsやmolucules、organismsの粒度が曖昧になってしまい混乱が生じました。

現在もフロントエンドのコンポーネント設計は模索中ですが、安易にAtomic Designを選ぶべきではないという教訓を得ました。

<details>

<summary>技術スタック</summary>

- Backend
  - PHP7.4 (https://www.php.net/)
  - Python3.6 (https://www.python.org/)
  - Laravel (https://laravel.com/)
  - PHPUnit (https://phpunit.de/)
  - SendGrid (https://sendgrid.kke.co.jp/)
  - Splunk (https://www.splunk.com/ja_jp)
- Frontend
  - JavaScript (https://developer.mozilla.org/en-US/docs/Web/JavaScript)
  - A-Frame (https://aframe.io/)
  - Vue2 (https://jp.vuejs.org/index.html)
  - Vuex (https://vuex.vuejs.org/ja/)
  - Jest (https://jestjs.io/ja/)
  - Sass (https://sass-lang.com/)
  - Maps JavaScript API (https://developers.google.com/maps/documentation/javascript?hl=ja)
  - Geocoding API (https://developers.google.com/maps/documentation/geocoding?hl=ja)
  - Places API for Web (https://developers.google.com/maps/documentation?hl=ja)
  - Figma (https://www.figma.com/)
- Infrastructure
  - MySQL (https://www.mysql.com/)
  - SQL Server (https://www.microsoft.com/ja-jp/sql-server/)
  - Azure (https://azure.microsoft.com/ja-jp/)
    - Virtual Machine (https://azure.microsoft.com/ja-jp/services/virtual-machines/)
    - Azure Functions (https://azure.microsoft.com/ja-jp/services/functions/)
    - Azure SQL Database (https://azure.microsoft.com/ja-jp/products/azure-sql/database/)
    - Azure Storage (https://azure.microsoft.com/ja-jp/product-categories/storage/)
    - Azure DevOps (https://azure.microsoft.com/ja-jp/services/devops/)
    - Traffic Manager (https://docs.microsoft.com/ja-jp/azure/traffic-manager/traffic-manager-overview)
    - Azure CDN (https://azure.microsoft.com/ja-jp/services/cdn/)
- DevOps
  - Circle CI (https://circleci.com/ja/)

</details>

## 株式会社スタッフサービス (https://www.staffservice-engineering.jp/)（エンジニアリング事業本部）（2018年7月〜2019年7月）

### 衛生画像の分析支援Webアプリケーション開発

#### 案件クライアント

大手日系メーカー子会社。

#### 期間

2018年10月〜2019年7月。

#### 開発内容

Spring BootでのバックエンドとAngularでのフロントエンド開発、開発後のテストで社内のみの総合試験を行いました。

開発では元々画面の描画完了まで10秒かかっていた処理を2秒まで短縮したこと、総合試験で不要ログ削除ツールのバグを発見するなどをしました。

また総合テストではテスト項目以外でも自分で考えてテストを行い、非常にクリティカルなバグを発見し報告しています。

<details>

<summary>技術スタック</summary>

- Backend
  - Java 8
  - Spring Boot
- Frontend
  - JavaScript
  - Angular 1.x

</details>

### 製薬開発支援Windowsアプリケーション開発

#### 案件クライアント

中小SIer。

#### 期間

2018年7月〜2019年9月。

#### 開発内容

C#でのWindowsアプリケーションの開発をしました。

実装前にテストケースのマトリックスを提出する必要があり、ここでソフトウェアテストの観点やテストの意義について学びました。

<details>

<summary>技術スタック</summary>

- Application
  - C#

</details>

## FIT株式会社 (https://www.fit4u.co.jp/)

#### 太陽光発電パネルの統計情報管理Webアプリケーション

#### 期間

2018年4月〜6月。

#### 開発内容

太陽光パネルを設置している顧客向けに発電量やCO2削減量などを確認できるWebアプリケーションを開発しました。

エンジニアとして初めての業務で、最初のリリースは4月中と決められていたため頑張って開発をして期日に間に合わせました。

PostgreSQL以外は初めてでしたが、業務中にキャッチアップを行って先輩に質問しながら開発を進めました。

リリース後はエンハンスでチャート画面の拡張などを行いました。

<details>

<summary>技術スタック</summary>

- Backend
  - C#
  - .NET MVC
- Frontend
  - Angular 1.x
- Infrastructure
  - PostgreSQL
  - IIS

</details>

# できること・得意なこと

## Backend

- 以下要素技術でのAPI開発
  - Laravel8
  - Node.js(TypeScript)
  - Python
  - Go
- テストコードでのunit testとintegration test
- playwright or puppeteerでのE2Eテスト

## Infrastructure

- RDBでの論理設計
- GCPでのインフラ構築
  - Cloud Run+Pub/Sub+Dataflow+Cloud SQL+BigQueryみたいな構成
  - GCLB+Serverless NEGも構築したことあり
- AWSでのインフラ構築
  - CloudFront+ALB+ECS on EC2+Auroraみたいな構成
  - SageMaker
- BigQueryでのデータ操作
- Athenaでのデータ操作
- TerraformでのIaC
- Jenkinsサーバ構築

## Frontend

- (React or Next.js) and TypeScriptでのフロントエンド開発
- jestを使ったテスト
- playwright or puppeteerなどでのE2Eテスト実装
- storybookでのカタログ作成とインタラクションテスト

## DevOop

- GitHub ActionsやCircleCIでのCI/CD構築
- renovate導入
- Docker & Docker Composeでの環境構築

# 性格

### 16personalities

管理者(ISTJ-A)
(https://www.16personalities.com/ja/istj%E5%9E%8B%E3%81%AE%E6%80%A7%E6%A0%BC)→
巨匠(ISTP-A)
(https://www.16personalities.com/ja/istp%E5%9E%8B%E3%81%AE%E6%80%A7%E6%A0%BC)–
論理学者(INTP-A)
(https://www.16personalities.com/ja/intp%E5%9E%8B%E3%81%AE%E6%80%A7%E6%A0%BC)

### DiSC

CSタイプ2。