職務経歴書

基本情報

項目名	内容
名前	菊地 弘祐（きくち こうすけ）
生年月	1984年3月
居住地	兵庫県西宮市
最終学歴	早稲田大学大学院創造理工学研究科建築学専攻博士課程単位取得退学

職務要約

ミッションクリティカル領域で、誤ったデータが「正として確定しない」ための設計・判断を担うバックエンドエンジニアです。会計データやポイント残高のような不可逆データを主戦場に、冪等性／状態遷移／再実行戦略／監査可能性を含む整合性設計を行ってきました。会員700万人規模のポイントプラットフォームでは、会計出力・外部カード連携・運用者向け処理まで横断し、短期的な性能最適化よりも長期運用での再現性・説明可能性・運用収束を優先して意思決定・実装しています。

技術領域

項目名	内容
Backend	Python（内部実装理解／メモリプロファイリング／非同期処理), Rust(限定的), Java（限定的）, Go(限定的)
Frontend	TypeScript(業務利用), Flutter(限定的)
Database	MySQL（主戦場：性能・運用）, DuckDB(業務利用), AWS Athena(業務利用)
Infra/Data	AWS(ECS, EKS, Lambda, CloudFront, CloudWatch, EventBridge, Step Function), Digdag（運用・トラブルシュート／実装解析)、 Embulk(運用・拡張／コードリーディング)
Others	Linux(プロファイリング／運用トラブルシュート), HULFT（レガシー理解）

得意領域

ミッションクリティカル領域のデータ整合性設計

- 会計データ／ポイント残高など、誤りが確定すると修正が困難な領域の設計・実装
- 冪等性、状態遷移、境界条件、再実行戦略を含めた「壊れない」設計
- 後から正しさを追試・説明できるデータモデル／処理構造の設計

長期運用を前提としたバックエンドアーキテクチャ

- 保守性・再利用性を重視した設計（ドメインモデル、クリーンアーキテクチャ等）
- 性能最適化よりも安全性を優先すべき局面の判断（止める判断を含む）

データ基盤・バッチ処理の運用設計

- バッチ処理／集計／検証の設計・運用（運用要件、異常検知、復旧を含む）
- ワークフロー（リトライ、通知、確定ルール）を含めた運用負荷の最適化

横断的な実装経験

- バックエンドを主軸としつつ、TypeScript + Vue.js によるフロントエンド開発経験
- UX／性能改善を目的としたフロント・バック両面での調整経験
- フロント／モバイルは専任が担当するが、障害調査時には Vue.js／TypeScript やアプリ側の状態遷移まで踏み込み因果を切り分け

高信頼運用とデリバリの両立（運用収束を含む）

- 期限制約下でも、冪等性／再実行性／監査性を落とさない実装・リリース設計
- レビュー待ちがボトルネックになる速度感で改善を回しつつ、境界（非同期・デプロイ・部分失敗・トランザクション境界）で壊れないことを優先

経歴

Engineer（YUZURIHA）

2021.01-現在 在籍

概要

- 大手スポーツ量販店向けポイントプラットフォーム（会員 700万人、将来 1000万人規模 を想定）
- ポイント残高・会計出力・外部カード連携など、誤りが確定すると修正困難な領域を担当
- ユーザー向けAPI〜管理画面/運用者向け処理まで横断し、データ整合性・運用性を含めて設計
- 冪等性/再実行、ロック順序、会計確定ルール等の設計規約を言語化してレビュー観点として定着させ、重大障害・誤計上に直結するリスクを設計是正で事前に潰し込み、運用で説明可能な形に収束させた

役割概要

プロダクト全体の技術方針の策定（チーム規模10名）

- メンバー10名程度の技術的な意思決定、アーキテクチャ設計の指針策定を担当。
- コードレビューや設計レビューの最終防衛ラインとしてプロダクトの信頼基準を担保。

特定プロジェクトの進捗管理や関係者調整（対象3-4名）

- サブチームのリーダーとして、タスクの切り出しや進捗管理、クライアントフェイシングなどを担当

採用活動の貢献

- エンジニア採用における一次面接を担当。主に技術力のジャッジを行う。

代表的な取り組み

プロダクト立ち上げ期における会計データ出力の不備

- 会計CSVの誤計上リスクをリリース前に特定し、計上ルールに整合する設計へ是正して未然防止

課題

会計システムが利用する CSV 出力について、特定条件下で正しく計上されない不具合が潜在していることを発見した。 当該 CSV は 月次集計に利用され、外部会計システムへ連携されるほか、複数部署が参照するデータであり、リリース後に不具合が発覚した場合、会計データの修正・再計上に多大なコストが発生するリスクがあった。

判断・行動

- 仕様と実装を突き合わせて検証した結果、ポイントの有効期限失効時に、会計上正しく計上されないケースが存在することを特定
- 計上ルール自体は正しい一方で、そのルールを前提とした設計になっていなかったことが根本原因であると判断
- 一時的な補正ではなく、計上ルールに合致する形へ設計を是正し、修正を実施

結果

- 月次・外部会計・複数部署参照という条件下で、会計データが誤った形で確定する事態をリリース前に未然防止
- 「システム上は正常だが、会計としては壊れている」状態をリリース前に排除

クレジットカード会社とのデータ連携（30万件／時間制約）

- 30万件/1時間SLAの外部連携を、ユーザー単位ロック＋検証分離＋冪等再実行で設計し、実運用で約6分の取り込みに収束

課題

クレジットカード会社とのデータ連携において、30万件のデータを1時間以内に Web アプリケーション内に安全に取り込むという制約があった。 誤投入や二重投入が発生すると、利用者情報やポイント残高が誤ったまま確定し、後追い修正が困難になるリスクがあった。

判断・行動

- 外部仕様がデータ番号の昇順に適用していけば良いという方針を先方から受け取ったが、そもそもの例外が多く、データ投入単位でまとまっていないため、そもそものルールの再定義を行った
- 並列処理の安全単位を確定するため、受領レコード単位（カード単位）からユーザー単位のキーを特定し、投入単位・ロック粒度・補正ロジックを設計した
- 大量取り込みと誤投入防止がトレードオフになるため、Validation と投入を分離し、投入条件を厳格化する判断をした
- またデータ投入時にエラーが発生することも考え冪等性を担保し、再実行時にもデータが壊れない構造を設計
- 取り込み途中でアカウント統合が発生する可能性を考慮し、ロックによって状態を確定させた上で投入（状態変更があった場合は適切にデータを補正）
- 結果として安全で実測で約6分での取り込みを実現

結果

- トランザクション整合性を厳密に要求される要件と高速かつ安全なデータ連携を実現
- 先方起因の障害10件（うち誤ったデータ送信3件）を投入段階で遮断し、不可逆な誤計上をゼロで運用
- 運用・再実行・障害時にも正しさを説明できる構造を確立

クレジットカード会社とのファイル連携基盤更新（HULFT／運用切替）

- OS/暗号更新を伴う高リスクなHULFT切替を、制約変化に応じて段階移行→一括移行へ方針転換し、監視・復旧手順込みで無停止に近い形で完遂した。

状況・課題

既存のクレジットカード会社とのファイル連携基盤について、 サーバーOS・通信アプリケーション・暗号方式の更新が必要となり、 連携停止時の業務影響が大きい高リスクな切替作業であった。 関係者は以下の4名で構成されていた。

- 自社：バックエンド担当（本人）、インフラ担当
- 先方：IT部門統括、インハウスカード事業マネージャー 当初はリスク低減のため、段階的な切替を前提に検討を進めていた。

判断・行動

- 更新対象となる OS／通信アプリケーション／暗号ソリューションを整理し、技術的な前提条件と制約を明確化
- ネットワーク構成・通信経路について、先方 IT 部門と詳細を詰め、段階的切替を前提とした進め方で一度合意
- その後、インハウスカード事業側から「クレジットカード会社側の切替コストが数百万円規模になる」という新たな制約が提示された
- コスト・リスク・復旧可能性を再評価し、段階切替を撤回して一括切替に方針転換する判断を実施
- 一括切替に備え、インフラ担当と連携してバックアップ・復旧手順を整理し、切替当日の重点監視体制を構築

結果

- 致命的な障害なく切替を完了し、ファイル連携を再開
- 外部関係者を含む意思決定と、リスクを取る判断を主導し、業務影響を最小化した形で連携基盤の更新を完遂

性能改善要求に対する設計判断（速度改善を止める判断）

- 危険な局所最適を止め、ボトルネックがMySQLのcommit fsyncであることを特定して、整合性を損なわない代替案（commit数削減・責務分離）へ意思決定を収束させた。

課題

性能改善要請があり特に現状調査も行われず同僚の案で進むことになった。ただ、現行構造のまま最適化を進めると、将来的にデータ整合性を破壊するリスクが高い状況だった。 この案を進めた最悪の場合、ポイント残高が誤ったまま確定し、取引履歴の再計算・復旧ロジックが二重化して原因追跡も困難になる懸念があった。

判断・行動

- これはポイント残高計算のロジックの根幹箇所で変更されなくなったデータを確定箇所であるが、そもそも実装難易度が高く理解できている人が限られている状態であった
- 実行が求められた primitive 化などの局所最適は、Python では cache locality があたらないため効果が低いどころか復旧ロジックの二重化・理解困難化を招くという仮説を立てた
- 調査の結果、ボトルネックがアプリケーションではなくMySQL の commit 時 fsync に待ちがあることを特定
- SWE としての実装継続では解決しないと結論づけ、現状のボトルネック調査した上でcommit 数削減や責務分離といった代替案を提示した上でタスクを PM に差し戻し

結果

- 長期的に破綻する可能性のある設計分岐を回避
- データ整合性を最優先とする判断を組織として合意

サービスリリース直後に発生したメモリリークの原因特定と再発防止

- 本番メモリリークの原因を永続キャッシュによるEngine参照に特定し、ライフサイクル設計（保持しない／破棄可能化）へ改修して安定稼働を回復した。

課題

サービスリリース直後には Web サーバーにメモリリークが顕在化し、本番運用に影響する不具合が発生していた。

判断・行動

- DB接続周りのライフサイクルを中心に調査し、functools.cache による永続キャッシュが Engine を参照し続けることでメモリが解放されないことを特定
- キャッシュ設計を見直し、重いオブジェクトを保持しない／必要なら明示的に破棄できる構造（clear/dispose）へ変更
- 以後の変更でも同種のリークを起こさないよう、責務とライフサイクル（生成・再利用・破棄）の方針を整理

結果

- メモリ使用量の継続増加を解消し、安定稼働を回復
- 本番運用でのトラブルシュートを“属人対応”ではなく“構造”に落とし込んだ

アプリリリース後に顕在化した DB デッドロックの原因特定と収束設計

- デッドロックの原因をロック取得順序の揺れに特定し、「A→B」に順序を規約化して恒久的に再発しない構造へ収束させた。

課題

アプリリリース後、本番環境で DB デッドロックが発生し、特定の処理が失敗、不安定な状態になっていた。障害対応として単発のリトライでは収束せず、継続的に発生する設計起因の問題と判断した。

判断・行動

- デッドロック対象の更新処理を特定し、複数テーブル（例：A / B）への行ロック取得順序がエントリポイントによって揺れていることを原因として切り分け
- テーブルAは常に更新対象でロックが必要である一方、テーブルBは条件により更新されるため、結果としての順序が混在し、相互待ちが発生していた
    - ある経路では「A → B」
        - 別経路では「B → A」

- 収束方針として、ロック順序を「A を必ず先に取得」へ統一し、B は必要な場合のみ後続で取得する設計に変更（ロック順序の規約化）

結果

- ロック取得順序の揺れを排除し、デッドロックを恒久的に解消
- 事象対応（リトライ・運用）ではなく、トランザクション設計の規約化により再発しない構造へ収束

レビュー・育成における取り組み

- レビューでは結論だけでなく判断材料（前提・リスク・代替案）を言語化し、チームの意思決定が再現可能になるように支援
- また、デプロイ時や非同期処理の境界でエラーやデータが壊れることがないかを重点的に指摘
- 感情的な指導や叱責は行わず、心理的安全性を前提としたフィードバックを重視
- 1on1 では、本人のスキルや状況を踏まえ、最も伸びる一点に絞って具体的な改善点を提示
- 日本語表現やドキュメントの書き方など、成果物の品質に直結する基礎スキルの支援も実施
- 個々の実装品質だけでなく、チームとしての判断基準やレビュー観点が揃うことを目的に、設計・実装の背景を言語化して共有
- 判断材料（前提・リスク・代替案）を言語化するレビューを徹底し、非同期境界やデプロイ境界で壊れない設計観点をチームの共通基準として定着させた。

その他プロジェクト

- クレジットカード会社とのファイル連携基盤の構築（HULFT／運用設計）
- 大手スポーツ量販店のポイントシステム刷新とその運用
- 大手スポーツ量販店の派遣人員募集サイトの作成
- 大手スポーツ量販店の派遣人員募集サイトの更新（リード）
- ポイントシステムのネイティブアプリ対応（サーバーサイド）

Data Reliability Engineer（Gunosy）

2019.05-2020.12 在籍

役割概要

大規模データ基盤において、アプリケーション・データ移行・集計・ビジネス運用を横断し、データの正当性・再現性・説明可能性を担保。

代表的な取り組み

SSP 収益データ取得・確定ワークフロー設計

- 速報値→確定値への特殊なfreeze仕様（60h/72h確定）をワークフローに落とし込み、Digdagの非直感的挙動をコードレベルで解析して再現性のある確定基盤を構築した。

業務内容

各メディアが利用する SSP の収益データを取得し、AWS Athena で分析可能な形に整備。最終的にはビジネスサイドで記事の配信元と按分する。

判断・行動

- 収益発生日から60時間後に速報値を確定し、72時間後に数字を確定しこれ以降は変更しないという特殊仕様に対応
- Digdag（Workflow Engine）の想定外の挙動に直面しながらも、実装をブラックボックス化せずコードを精査して原因を特定・解決

成果

- ビジネス要件とデータ整合性を両立したワークフローを構築
- 予期せぬ挙動に対しても再現性をもって対処できる基盤を確立
    - 参考： https://tech.gunosy.io/entry/escape-from-pitfall

- この際にビジネスで収益按分割合の誤りを指摘

データ基盤民主化に向けたワークフローテンプレート作成

- 特定チーム依存だったワークフロー基盤をテンプレート化し、責任分解を明確にして安全に再利用可能な形へ抽象化した。

業務内容

特定チームのみが利用していたワークフローエンジンを、他チームでも安全に使えるようにするためのテンプレートを設計。

判断・行動

- bash による最低限の入力で利用可能なテンプレートを作成
- 利用者側と基盤側の責任分解点を明確に定義

成果

- ワークフロー作成・運用コストを大幅に削減
- 基盤チーム・利用チーム双方の負担を軽減

運用負荷低減を目的としたワークフロー整理

- 「時間経過で自然収束するエラー」を構造化し、通知設計を再設計して障害対応を人の判断からワークフロー構造へ移行した。

業務内容

エラー通知が「データ欠損防止」という観点で設計されておらず、不要な運用負荷が発生していた。

判断・行動

- 時間経過により自然に正しい状態へ収束するパターンを分類
- 入力・出力別に「次セッションで必ず正しくなる」構造へワークフローを再設計

成果

- データ正当性を維持しつつ、運用負荷を体系的に削減
- 障害対応を“人の判断”から“構造”へ移行

その他プロジェクト

- オンライン機械学習基盤構築
- 特許データの前処理と検索基盤の構築
- データマート作成と保守

システムエンジニア（パーソルキャリア）

2017.10-2019.04 在籍

概要

- 求人・転職領域の大規模Webサービスおよびデータ基盤関連プロジェクトに従事
- 既存システムの保守・改善を中心に、安定運用を前提としたバックエンド開発を担当
- データ基盤やバッチ処理における前提条件のずれや検証不足が、後工程で大きな修正コストを生む場面を多数経験
- この経験を通じて、実装そのものよりも「設計段階での判断」や「正しさを説明できる構造」の重要性を強く意識するようになった

プロジェクト

- 機械学習システム用のオンプレミスHadoop基盤構築
- 機械学習プロジェクトの進め方に関するドキュメント整備
- 既存機械学習エンジンの改修とそのレビュー
- 事業サイドをターゲットにした基礎統計学勉強会

データマイニングエンジニア（株式会社IPオンウェブジャパン）

2016.02-2017.09 在籍

概要

- 広告配信・収益データを対象とした分析および運用支援に従事
- ログや集計結果をもとに、数値の不整合や想定外の挙動を発見・原因特定する役割を担当
- 「システム上は正常に見えるが、データとしてはおかしい」状態を継続的に扱う
- データの変化や分布の違和感から異常を見抜く経験を積み、後工程での修正が困難になる前に問題を潰す姿勢が形成された

プロジェクト

- 多腕バンディット法を用いたキャンペーン予算最適化
- 海外SSPのPrivate Auctionの導入
- インフィード広告の配信トラブル改善
- 特定代理店に対するパフォーマンス向上コンサルティング

エンジニア（株式会社Spotlight）

2014.04-2016.01 在籍

概要

- 主に楽天チェックというO2O送客サービスに携わった
- レポート生成・表示処理の速度改善に取り組み、ユーザー体験と運用効率の向上を実現
- 特定の技術領域に閉じず、課題解決に必要な技術を分け隔てなく扱う姿勢で開発に従事
- この経験を通じて、現在のフルスタックな視点と、技術選定を手段として捉える考え方の基礎が形成された

プロジェクト

- 社内レポートシステムの作成
- テストユーザー機能追加
- 大手コンビニチェーン店導入のための新規機能開発
- サービスのパフォーマンス改善

スキルなど

資格

IELTS Academic 6.5 2012.11

受賞・その他

- 早稲田大学EDGE グローバルアントレプレナー育成プログラム修了 2016.12
- The Fab Academy, Certificate 2013.08
- 日本建築学会関東支部若手優秀研究報告賞 2013.03
- 早稲田大学鶴田奨学金 2013.02
- 早稲田大学博士実践特論カリキュラム修了 2011.09
- Machine Learning, Coursera(Stanford University) 2022.02
- Deep Learning, Coursera(Deeplearning.ai) 2022.03

論文

- Location Based Social Network による都市分析 都市・建築分野でのビッグデータ解析の試み, 2013年度日本建築学会全国大会（北海道）情報システム技術部門研究協議会　寄稿文

- Collective Background Extraction for Station Market Area by using Location Based Social Network, Journal of Civil Engineering and Architecture, Vol. 7(3), 282-289, 2013
- スマートライフ, パレード社, 2011
- A New Method for Analyzing the Relationship between City and Human Behavior using Geo-Tagging Social Networking Service, International Union of Architects Academic Program, 10074, 388-393, 2011
- Arduino・Pachube・SketchUpと連携した建築モニタリングシステムの基礎的研究, 日本建築学会技術報告集, Vol. 16, No. 33, 791-794, 2010

技術アウトプット

- https://http3-explained.haxx.se/ja
  - Daniel Steinberg 氏が書かれた HTTP/3 (QUIC) の解説記事を一部日本語訳
- https://qiita.com/kikuchi-yzrh/items/56b2368e4473e19ec1d9
  - cache locality が性能に与える影響を整理し、最適化が効く条件／効かない条件を言語化
- https://qiita.com/kikuchi-yzrh/items/9680fd85f0dcb474ef18
  - SQLAlchemy の IdentityMap をコードから追い、同一性・更新・参照が破綻する境界条件を整理
- https://qiita.com/kikuchi-yzrh/items/baaab16cc1258e81e476
  - MySQL の全文検索（n-gram）で “commit しないと反映されない” 仕様を検証し、テスト時の落とし穴を整理
- https://tech.gunosy.io/entry/escape-from-pitfall
  - digdag の retry でタスクが重複実行される落とし穴を解析し、冪等性と再実行設計で回避策を確立

作品

Plant Pod 2013.03

植物と人間はお互いのコミュニケーションチャンネルが異なるので、コミュニケーションすることは不可能である。しかしながら、植物の生育環境をセンシングすることで、その植物がどうしてほしいのかを検知することはできる。この状態を顔文字として表現し、人間に対して察することを促すことで、コミュニケーションを取らせることを目的としている。
https://www.youtube.com/watch?v=IEh0-y3y_0s

Live Camera 2010.08

プライバシーを保護しつつも研究室の内部の状態を可視化することを目的としている。この実現のために、Webカメラの色彩情報をクラスタリングをした画像を保存し、この時系列データをマウスのカーソルに合わせて表示させた。誰がいるのかは所属者のみがわかる程度に抽象化されているために、この情報をみて研究室に来る人が増え、コミュニケーションが増加した。 https://www.youtube.com/watch?v=WbKnb9Cbi8s

Environmental Sensing 2010.03

センサーを大量にばらまくために、格安のセンサーモデュールを開発した。この情報をサーバーと同期させ、3DCAD上のオブジェクトに反映させた。これにより、居住者も設計者も住宅の状態を把握することができた。
https://www.youtube.com/watch?v=4_mtrW7d3nc