📄 職務経歴書

👤 基本情報

項目	内容
氏名	三好 翔太
生年月	1994年6月
居住地	神奈川県
最終学歴	千葉大学 文学部 行動科学科

💼 職務経歴詳細

🏢 @東急株式会社

項目	内容
雇用形態	正社員
在籍期間	2025年3月 ～ 現在
ステータス	🚧 coming soon

🏥 医療者/製薬企業向け Web サービスの機能強化・保守運用等 @Dr.JOY株式会社

> **製品概要:**
> 医療機関の「働き方改革」を支援するコミュニケーションインフラ。勤怠管理システムをはじめ、業者管理機能や地域連携機能などの機能を持つ。医療機関と製薬企業間のコミュニケーションを円滑化する役割も担っている。

項目	内容
雇用形態	正社員
在籍期間	2020年2月 〜 2025年2月
チーム規模	PM/PjM: 1人、BE: 1人（私）、FE: 1〜3人（ベトナム）

🛠️ 技術スタック

カテゴリ	技術
言語・フレームワーク	`Java` , `Spring Boot` , `TypeScript` , `AngularJS` , `Go`
データベース	`MongoDB` , `PostgreSQL` , `BigQuery`
インフラ・クラウド	`GKE` , `Cloud Pub/Sub` , `Firebase`
BI・分析	`Metabase` , `Looker`
開発・運用	`Git` , `IntelliJ` , `CircleCI` , `DroneCI` , `gRPC` , `Docker` , `JUnit`

🎯 主な実績・取り組み

💳 1. サービス有料化に際して Stripe を導入し、サブスクリプションを管理する基盤を整えた

🔍 **課題:**

クレジットカード支払いによるサブスクリプション方式でのサービス有料化を行うにあたり、決済機能を具備する必要があった。

🛠️ **施策:**

1. 決済管理 SaaS として Stripe を導入した
    1. Zuora, GMO Payment Gateway と比較検討して、費用面とサブスクリプション設定の柔軟さから Stripe に決定した

2. ビジネスサイドメンバーとサブスクリプションのサイクルや料金について擦り合わせを繰り返し、Stripe 上での料金・サブスクリプション設定を決定
3. **Stripe API を利用して決済基盤を実装した**
4. 運用メンバーとの擦り合わせを行い決済失敗時の対応フロー等を策定した
    1. 決済失敗者（要対応者）情報を自動で収集するために SpreadSheet + GAS で仕組み化を行った

⚡ 2. 勤怠機能の計算コアモジュールを改善可能な状態にした

🔍 **課題:**

参画時点で勤怠機能の計算コアモジュール周りは下記のような状況であった。

1. **仕様を把握しているメンバーが不在**（設計・実装者は退職済み）
2. 月間の出勤簿画面上から実際に1ヶ月分の勤務予定・実績・残業申請等を入力することでしか計算結果を確認できず、**修正→テスト結果確認のサイクルが低速**

その中で下記のような対応が必要とされたが、リグレッションの発生を検知したり影響範囲を特定したりすることが困難な状況であった。

1. 運用中のユーザから計算誤りを指摘された際の修正対応
2. 裁量労働制、変形労働制等への対応

🛠️ **施策:**

1. **UnitTest の整備**
    - 現状の挙動を把握するためにパブリックな計算メソッドに対して JUnit でテストコードを付加した
        - → **リグレッションを検知可能になった**
2. **ビジネスサイドメンバーへの確認依頼フォーマットを策定**
    - 上記 UnitTest 実行時、勤務予定・実績・申請等の条件と計算結果を統一フォーマットで標準出力へ吐き出すようにした
        - → **SpreadSheet に貼り付けることでビジネスサイドのメンバーへの確認依頼をスムーズに行うことができるようになった**

📊 3. PDF 帳票出力用の共通モジュールを設計・実装した

🔍 **課題:**

Dr.JOYでは勤怠管理サービス上で Excel 形式の帳票を出力できる機能がすでに具備されていた。 しかし病院側ですでに使用している帳票フォーマットに則って PDF 形式で出力したいという要望が上がっており、 ナイーブに設計すると導入施設が増えるたびに PDF フォーマットを作成する必要があり、弊社のエンジニアリソースでは不足してしまうと考えられた。

🛠️ **施策:**

1. **SVF という帳票作成 SaaS を導入した**
    - 帳票デザインを非エンジニアでも作成でき、コンポーネント化することで導入施設ごとのマイナーチェンジへの対応も容易

2. **SVF API に勤怠データを投入する Interface を実装した**
    - 投入する各勤怠データ計算を実装するのはベトナム子会社のエンジニアになる予定であり、彼ら/彼女らが SVF 側の事情を考慮しなくて済むようにモジュールを作成した
    - 全く異なる種類の帳票を実装する場合でも、`Marker Interface`を実装した`Bean`を作成し必要なフィールドを定義するだけでデータを SVF API に投入できるよう実装した
        - 勤務管理簿、休暇管理簿等、etc

📈 4. BI ツールの運用改善を行った

🔍 **課題:**

前任者の退職に伴い、Metabase という BI ツールの運用を引き継いだ。Metabase は GKE 上に構成されており、データソースとしては PostgreSQL を利用していた。

サービスで利用している MongoDB から BI 用の PostgreSQL への ETL Job も GKE 上で実行されていたが、下記のような課題が発生していた。

1. 毎日全レコードを対象にしていたため、データ量の多いコレクションでは**頻繁にタイムアウトが発生**していた
2. 上記に伴い実行時間が非常に長くなっており（**10時間程度**）深夜に実行開始しても翌日の業務開始までに処理が完了せず、ビジネスサイドのメンバーの業務に支障をきたしていた
3. ETL が失敗した場合に個別にコレクションをリカバリしたり、PostgreSQL のパラメータを調整したりする作業が頻繁に発生し、私の業務を逼迫しつつあった

🛠️ **Metabase 改善に関する施策:**

1. **圧倒的にデータ量が多い勤怠サービス関連のコレクションを別個の ETL で処理するように変更した**
2. その中でも特にデータ量が多く頻繁にタイムアウトしていたコレクションについては**3分割して投入するように変更した**
    - ビジネスサイドのメンバーが主に必要とするのは勤怠サービスのデータ以外の部分であったため、彼ら/彼女らの業務への支障を取り除くことはできた

⚡ 5. アクティブユーザデータ集計ツールの速度改善と手順の簡素化を行った

🔍 **課題:**

前任者の退職に伴い、月次のアクティブユーザのデータを集計する業務を引き継いだ。前任者が作成した集計ツールを利用して作業するが、下記のような状況のため他の業務に支障をきたしていた。

- **ローカル PC の CPU をほぼ 100% 使い切ってしまう**
- **4時間程度処理に時間がかかる**（ローカル DB へのデータインポート2時間、集計処理2時間）

また、集計ツールから出力された tsv ファイルをビジネスサイドのメンバーが SpreadSheet + GAS で加工して最終的な成果物を作成する手順となっており、彼の業務も逼迫していた。 さらに、ユーザー数の増加に伴い、SpreadSheet で扱えるレコード数の限界を迎えつつあり、早急に対応が必要であった。

🛠️ **施策:**

1. **余分なデータをダウンロードしていた箇所を取り除いた**

2. 集計ごとに動的に JavaScript を生成して実行する形だった集計処理部分を **Go で再実装し、コマンドラインツールのバイナリとして配布可能な形にした**
3. **MongoDB のクエリのパフォーマンスを改善した**
    - `lookup` の処理がボトルネックとなっている部分が多かった。そこでメモリ利用量が増えることは承知の上、集計アプリケーション側で `join` 相当の処理を行うように変更した
4. **コマンドラインツールで最終成果物まで作成するようにした**
    - テスト可能性、及び他のメンバーが個人で作成していたツールとの互換性のためにこれまで出していた tsv ファイルも中間成果物として出力するようにした
5. **実行環境をコンテナ化し `make` コマンドだけで作業を完結**できるようにし、作業の引き継ぎを容易にした

> 📊 **成果:** 処理時間が **4時間 → 2時間（50%削減）** に短縮！（ローカル DB へのデータインポート1.5時間、集計処理10分）

💻 グループウェアシステムの新機能開発・不具合修正 @ワークスアプリケーションズ

> **製品概要:**
> Ariel AirOne Enterprise for Companyというオンプレのサーバにインストールして利用するグループウェア。スケジュール・タスク管理やファイル管理等グループウェアとしての必須機能はもちろん、ワークフロー等のモジュールをユーザ側でカスタマイズできる柔軟性が特徴。
> **プロジェクト概要:**
> PG としてスケジュール管理・タスク管理・ファイル管理モジュールを中心に、機能追加・不具合修正を実施した。

項目	内容
雇用形態	正社員
在籍期間	2017年4月 ～ 2019年7月
チーム規模	PG: 8～10人（私）、QA: 4～10人、CS: 10～15人、コンサル: 20～25人

🛠️ 技術スタック

カテゴリ	技術
言語・フレームワーク	`Java` , `Spring Framework` , `JavaScript`
データベース	`Oracle 11g`
開発・運用	`Subversion` , `IntelliJ` , `Jenkins` , `Junit`

🎯 主な実績・取り組み

🕒 1. タイムゾーン問題を改善

🔍 **課題:**

Ariel AirOne Enterprise という製品には次の4つのタイムゾーンの概念が別々に存在していた。さらに機能ごとにこれらの適用優先度がまちまちであるため、時刻周りの不具合が多発してしまっていた。

1. オンプレサーバに導入するタイミングで設定されるタイムゾーン
2. 管理者アカウントで設定できる、テナント単位のタイムゾーン
3. 一般アカウントで設定できる、個人単位のタイムゾーン
4. スケジュール・タスク機能において、各スケジュール・タスクリソースに割り当てられるタイムゾーン

🛠️ **施策:**

1. 時刻編集時・表示時の不具合がユーザビリティを大きく損なうと思われる、**スケジュール・タスク機能にまずはスコープを絞って**不具合チケットを収集した
2. 各不具合チケットについてそもそものあるべき姿が曖昧だったので、各チームの古株のメンバーに**ヒアリング・合意形成しながら仕様として明文化した**
3. 明文化した仕様に沿ってスケジュール・タスク機能の主要なユースケースに対して **JUnit でテストコードを書き、マネージャにレビューをしてもらいつつ修正方針について合意形成した**
4. **QA メンバーにリグレッションテストをしてもらいながら**、JUnit のテストコードを Pass するように修正を実施した

🎯 2. タスク機能のユーザビリティ、及びスケジュール機能との連携を改善

🔍 **課題:**

Ariel AirOne Enterprise という製品には数多くのグループウェア機能が存在するが、中でも**スケジュール機能の利用率が高かった**。次いでタスク機能の利用率が高かったが、これらの機能間の連携は一切なく、そのためスケジュール機能をメインで使うユーザはタスク機能をあまり使わない傾向にあった。

またタスク機能は利用率の高い機能でありながら、ユーザビリティの面での課題が挙がっていた。

1. **タスクの進捗状況を更新する方法が UI 上から分かり辛い**。タスク機能の一つ一つの「タスク」リソースは開始前、着手中、完了、などの進捗ステータスを持つが、ユーザがこれを変更するための操作の動線が不適切な階層に配置されていた

2. **タスクを一覧画面から検索・フィルタリングする術がない**。完了タスクが溜まっているヘビーユーザーほど使い勝手が悪化してしまっていた

🛠️ **施策:**

1. **タスクの進捗ステータスを更新するためのボタンをタスクの詳細画面上に設けた**。ボタンのデザインはスケジュール機能上の出欠表明のボタンに合わせて、スケジュール機能との連携の布石とした
2. **タスク一覧において、進捗ステータスで表示するタスクをフィルタリングできるようにした**
    - よりリッチな検索等を実装する手もあったが、進捗ステータスでのフィルタリングは後述のスケジュール連携時にも利用できるため、こちらを採用した
3. **期日を設定したタスクについてはスケジュール画面にも表示できる機能を実装した**
    - プロジェクトにデザイナーがいないため、スケジュール画面に表示する際のアイコンも自分で作成した
4. **スケジュール画面に表示させるタスクを進捗ステータスでフィルタリングできる機能を実装した**
    - 社内でのドッグフーディングのフィードバックから、デフォルトでは完了ステータスのタスクを表示させないように修正した