経歴詳細

2019年12月〜現在　株式会社 sorich（ソリック）

入社目的

- 準委任契約・受託契約のため、自社サービスに近い環境で、開発を行えること。
- 社内で様々なプロジェクトに携われて、フロント・サーバーサイドの力がつく。
- 新規開発において、要件定義→設計→実装→テストの経験を積めること。

2019年12月~ 2020年04月 宿泊管理システムの開発

規模	体制	担当	フレームワーク・言語
5人	Engineer leader 1人 backend 2人 frontend 2人	backend	Laravel、 PHP

獲得した技術・経験

- フレームワーク乗り換え時のキャッチアップ
- Laravel
    - Laravelのコレクション操作、クエリビルダ、ハッシュ、暗号化等の知識
    - クエリビルダをSQLに起こして、取得したいデータの加工および獲得
- API作成
    - 要件定義・設計を基にしたswaggerの設計
    - swaggerを用いたWEBAPIの作成
    - 基本的なHTTPメソッドの実装
        - 基本的なステータスコードの知識
    - トランザクション、例外処理の実装
- デバッグ、改修
    - 関数、ツールを用いたデバッグ
    - curlコマンド, Postmanを用いたAPI開発
    - chrome developer_toolsを用いたフロントエンドの改修
- データベース
    - 10カラム以下のテーブルの論理設計
        - テーブル分割や正規化の知識
- テスト
    - fakerによるテストデータの作成
    - 異常系も含めたテスト仕様書の作成
    - 同地分析、デシジョンテーブルなどを用いたテストの実施
    - 総当たり攻撃などの脆弱性に関するテスト
- チーム開発における基本的なコミュケーション
    - わからない・知らないことは、理解した時点まで共有して聞く。
    - 要件が曖昧である場合は、チーム・クライアントサイドに相談し、確定させる。
    - タスク遅延が発生する見込みがある場合にマネージャーに都度相談を行う。

- チーム開発の良さ
    - 一人では達成できないプロジェクトに取り組める。
    - 一時的に期限が厳しい環境だとより一丸となれる。

開発終盤の開発工数例

要件確認・設計	実装	レビュー修正
0〜1人日	1〜3人日	0〜1人日

環境

要件定義・設計段階を終えた段階でプロジェクトにアサインされました。クライアントさんが予算と工数をかなりキツく出してこられて、追加機能も追加工数なしで対応される環境でした。必然的に、毎日、23時まで残業することになりましたが、チーム一体で仕事ができていて成長・充実感がありました。

過程

入社直後に、配置転換があり、Railsではなく、Laravel,Vue.jsのプロジェクトに配属されました。3週間のLaravel, PHPのキャッチアップの時間をいただき、実務に取り組みました。 最初に、在庫管理やユーザ管理の機能を担当しました。慣れてくると日次集計表のwebAPIを開発しました。また、DBの論理設計も少し任せていただけました。最後に、システム全体のテスト仕様書の作成、実施を担当いたしました。

改善

webAPIの開発経験がなかったために、webを支える技術やwebAPIなどの書籍で体系的に習得しました。加えて、HTTPのステータスコードの意味や例外処理を学びました。クエリビルダに関しては、リファレンスを読み、手元でSQLを確認しました。加えて、達人に学ぶシリーズを読み、SQL、DB設計の知識を深めました。テスト仕様書に関しては、演習形式の書籍で、同値分析、境界値分析、デシジョンテーブルなどを網羅的に学びました。

振り返り

新しい環境、新しい言語で開発することになり、とても不安でした。毎日11時まで必死に頑張ったため、swaggerやHTTPメソッド、DBの関係を考慮しながらのクエリの作り方やDB設計など多くのことを吸収できました。特に日次集計表では、ポイント、クレカ、現金の扱いで表示する金額が異なってくるので、仕様整理に苦労したのを覚えています。最後に、テスト工程に関しては、私1人にテスト仕様書の作成を任せていただき、自信がつきました。

2020年04月~8月 ECサービスの申し込みフォーム作成 (フルリモート)

規模	体制	担当	フレームワーク・言語
3人	Engineer leader 1人 frontend 2人	frontend	Vue.js、Javascript

獲得した技術・経験

- Vue.js
    - ディレクティブ、算出プロパティ、ライフサイクルフックなど基本的な知識
    - input、buttonなどのデータバインディング、要素の操作
    - コンポーネントに関する知識
        - 単一コンポーネント (呼び出しや値の受け渡し)
- 失敗して体得したフルリモートワーク下のタスクの進め方
    - 進捗、タスク管理を出社以上により密に行う。
    - 疑問が浮かんだ場合は、要点をまとめ、すぐ相談する。（時間損失を考慮）

過程

初めてのフロントの担当でした。最初の2ヶ月は、Vue.jsの書籍を用いて写経して動かして学びました。その後、実務に入るも、全く作業が進まずに、遅延が発生しました。原因としては、上長は忙しくて些細なことで相談するのが、迷惑だと思い込んでいたことでした。最後に、上長へ相談し、巻き取っていただきました。

改善

このプロジェクト以降フルリモートであっても、すぐにチャットや電話で相談することを意識いたしました。

振り返り

コロナ禍で全社完全フルリモートが始まった時期でした。上長の時間も大切ですが、自分が稼働しないことによって会社に被る時間・賃金損失も考慮すべきでした。Vue RouterやVuex、Jestなどのテスト自動化に関する知識も後に学びました。

2020月09月~11月 継続課金システムの改修 (フルリモート)

規模	体制	担当	フレームワーク・言語
4人	Engineer leader 1人 backend 3人 frontend 1人	backend frontend	Codeigniter、PHP JQuery、Javascript

獲得した技術・経験

- PHP
    - 公式リファレンスで基本的な文法
        - base64（暗号化）、継承、配列操作、時間操作
    - 独習PHPで網羅的に習得
- CodeIgniter
    - Laravelと同様に基本的な機能の習得

        - ルーティング、MVC、クエリビルダ、session、cookie
        - twigなどの独自テンプレート
- 更新が止まってしまったフレームワークの改修
    - 海外の英語資料を読解して課題解決
- リーダブルコードとリファクタリング
    - 関数名や命名規則に関するアンチパターン
    - 関数や変数の集約
    - 後任者が改修しやすいコードの実装
- オブジェクト指向設計
    - クラスにおける抽象、具象の概念
- JQuery
    - idやclass指定で値取得、操作
    - イベントハンドラの実行タイミングの把握
    - HTMLタグやCSSのstyleの修正など知識
    - Ajaxを用いたAPIの非同期通信

過程

CodeIgniterを使用していましたが、素のPHPで実装されている箇所が多く、変数名がuserPublicProfileなのに、UserModelだけの情報を扱っていたりて、システムのコードを読み解くのに苦労しました。また、コピペ流用されたコードが大量に散見されて、抽象化を意識してリファクタリングを行いました。フロント側では、JQueryで画面に表示するコンテンツの動的な制御を改修しました。

改善

PHPリファレンスや書籍を参考に、基本文法を手元で確認して身につけました。変数名の問題もあったので、リーダブルコードを読み、実践しました。JQueryは、1つずつ実行したいことを調べて、画面で確認し、実装しました。抽象化に関しては、オブジェクト指向やリファクタリングの書籍通して概念を学び、取り入れました。

振り返り

プロジェクト自体は、クローズ期で短かったですが、良い経験が積め、成長を感じました。 オブジェクト指向やリファクタリングは、取っつきにくいイメージがあったのですが、実際に、学んでみると全体を理解しやすくするために、シンプルにするという考え方に共感できました。一度では、理解しきれなかったので、時折読み返しています。JQueryに関しては、DOMを直接操作するというのが理解しやすく、Vueを初めて触った時と比べ、すぐに習得できたのが嬉しかったです。

2020年12月~2021年12月（現在）送金・決済アプリの開発 (フルリモート)

規模	体制	担当	フレームワーク・言語
20人	VPoE 1人 Infra 1人 Engineer leader 3人 Bizdev 5人 backend 9人 frontend 2人	backend	Ruby on Rails、Ruby

獲得した技術・経験

- Rails (TrailBlazer)
    - s3からcsv読込をして夜間定期実行するバッチの実装
    - ymlを用いてのawsのcron設定方法
    - ログイン処理の総当り攻撃の対策（アカウントロック）
    - fileUpload等を用いたディレクリトリトラーバサルの対策
    - Rubyのバージョンアップ作業
- Rspec
    - 異常系も含めたテストコードの実装
    - テストがテストになっているか
    - 必要最低限、重要な点はテストできているか
- エラーハンドリング
    - 204,400,404,500など、適切なエラーを表示し、処理する。
- 小さいプロジェクトを1人で担当
    - 顧客折衝 (進捗管理、期限の交渉)
    - 既存の機能のリプレイスの見積もり、タスクの細分化
    - 実装後のやりとりし、追加の改修や要望の対応
- 運用・問い合わせ
    - システムに関わる必須事項のドキュメント化
    - 過去ドキュメントやチャットでタスク実施
    - CSなどの非エンジニアとのコミュニケーション
        - 詳細な聞き取り、課題の整理
            - 正確な発生日時、エラーメッセージ、OS、端末情報、キャプチャ
- 障害対応
    - 正確な情報の共有
    - 障害発生時はすぐに報告。
        - CS等と連携し、対応を行う。

開発の開発工数例

1機能、1APIを作成する際の平均的な工数

要件確認・設計	実装	テストコード実装	レビュー修正・リリース
0〜1人日	2〜4人日	1〜2人日	1人日

過程

最初に取り組んだのは、3万行のcsvデータを差分確認し、登録、削除、更新対象抽出するバッチでした。次は、地方自体専用フォームの大改修を担当しました。開発期間が3ヶ月あったので、タスクを大まかに見積もりし、分割し、実装をしました。直前に、トラブルに見舞われましたが、無事リリースできました。現在は、新規開発のwebAPIを担当を行っています。

改善

ドキュメントが足りておらず、上長に都度仕様をテレカンで確認し、自らドキュメント化しました。デバッグしても原因がわからない事象に当たったときは、すぐにメンバーに共有し、対応策を考えるようにしました。定期バッチ

は、バッチ中にメモリエラーが起こしてしまうので、allなどですべてを取得しないように、必要分の取得できるように改良を行いました。他に、N+1が発生している箇所をbulletで調査して、対応しました。また、設計・要件定義のタスクを任された際に、見積もりの精度が低くかつ課題の設定や、タスクの細分化の力が弱いと感じたので、関連する書籍を購入し、実務に取り入れました。

振り返り

trailblazerというRailsのgemを使っており、Railsとは違う独特の書き方で戸惑いました。アプリ規模が大きく、リポジトリ数も10を超える巨大サービスで、全体を把握している方は 1人もおらず、チームメンバーと知識を共有しながら、タスクに取り組みました。 リリース前に、minimagickのiphoneの画像拡張子heicが登録できない事象に見舞われたときは、ギリギリまで調査したのですが、解決できませんでした。最終的に、Rubyのバージョンを上げて改善できたのは良い教訓です。見積もり・要件定義に関しては、知識を深められましたが、経験値が足りてないのを痛感しました。これから経験を積んで、自分なりの方法を確立していきたいです。

2019年4月〜10月　株式会社ジラフ

入社目的

- ユーザーと近いWEBサービスの開発を行いたい。
- Railsでプログラミングしてみたい。
- 以上の2点でエンジニアで入社しました。

匿名質問サービスの改修　2019年04月~10月

規模	体制	担当	フレームワーク・言語
10人	Engineer leader 2人 Bizdev 2人 backend 3人 frontend 2人	backend	Ruby on Rails、Ruby

獲得した技術・経験

- 公式リファレンスをもとにした技術習得
- Ruby公式リファレンスで各クラスの組み込みメソッドの知識
- 既存システムのコードリーディング方法(API仕様書、DB設計から確認する。)
- 基本的なGUI形式のDBの操作方法
- chromeの検証機能を用いた画面側のデバッグ
    - network項目の見方やconsole、Elementの使い方
- エンジニア、ビジネスの両視点で機能開発が必要かの議論
- Yes, No を意識したや目的を持った質問方法

過程

入社前にRailsチュートリアルを取り組んだものの、研修無しで2日目から配属、実務で1ヶ月はシステムのコードが読めず、デバッグ方法やRuby標準の組み込みメソッドすら知りませんでした。当然ですが、業務の遅れが発生しました。しかし、当時は、遅延に対する恐怖心によって、相談できませんでした。

改善

まずは、デバッグ方法を教えていただいて、変数の中身やクラスについてリファレンスをもとに学びました。また、Rubyリファレンスの標準の組み込みメソッドをクラス順に1つ1つ実行して、エンジニア定例で発表しました。加えて、直属の先輩に、報連相をするようにしました。

振り返り

早く仕事できるようになって、成果を出さなければという強迫観念にとらわれていて、空回りしていました。もっと早いうちにSOSを出して地道に1つ1つできることを増やしていけたら成長スピードが変わったたと思います。

1社目 退職理由

当時、社員数100名に迫る急拡大をし、その後業績が悪化。人員整理が始まり、転職せざるをえなくなってしまったため。（現在は、40名程度）

現職ではなく、転職活動に踏み出した理由

- プロジェクトの技術選定は、社長の裁量で決められ、配属（技術）の選択権限がない。
- 開発期間が短期〜中期であることが多く、技術が深化できない。
- クライアント、ユーザからも機能開発しても、フィードバックがない。
- 社内で技術的フィードバックがないので、もっとボコボコにコードレビューされたり、ペアプロを通して成長したい。
- 他言語で、特に静的型付け言語（JavaかTypescript）に挑戦したい。
- 現状、クライアントから決められた仕様を開発する契約になっている。

現職での改善行動

- 上長・社長に挑戦したい技術、プロジェクト、今後のキャリアを相談するも、社内的に配置転換できないと伝えられる。
- Typescriptの学習を始め、簡単なアプリ制作を行い、習得を図る。