chocoZAP

1年間の高速化とリファクタリングの実践

RIZAP GROUP TECHNOLOGIES

DEC 2024,

新井 幸希

Arai Koki

RIZAPテクノロジーズ株式会社

プロダクト開発統括1部

chocoZAPバックエンド担当

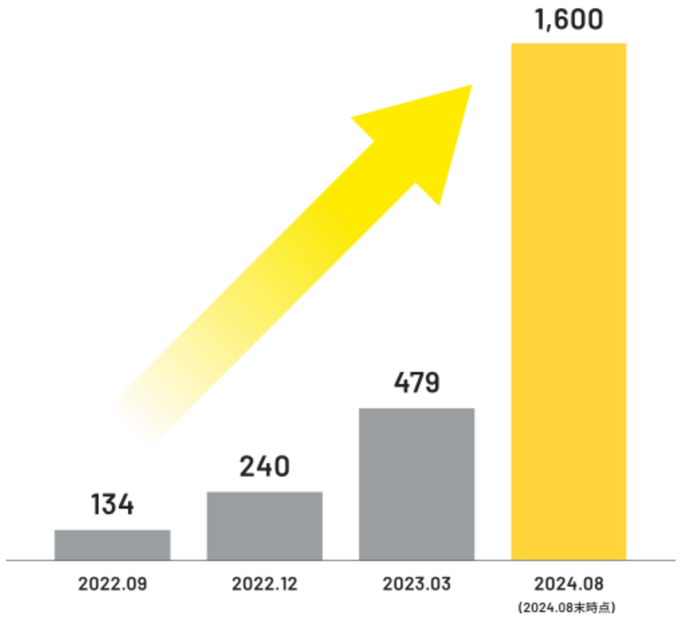
趣味・サッカー & 格闘ゲーム

chocoZAP

全国1,600店舗以上

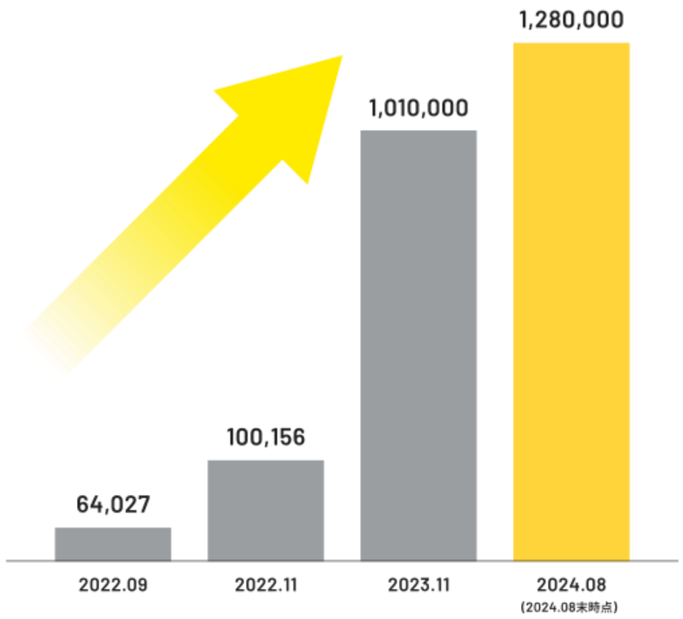
(2024.08末時点)

※2023年6月以前の数値は、chocoZAP以外の旧屋号を含む合計店舗数



会員数128万人突破

(2024.08末時点)



2026年3月までに2800店舗へ

中期経営計画の達成に向け
chocoZAP出店を継続

chocoZAPアプリの使用言語

chocoZAPが

抱えていた課題

chocoZAPが抱えていた課題

chocoZAPが抱えていた課題

01 | リクエストレスポンスの速度が遅い

chocoZAPが抱えていた課題

01 リクエストレスポンスの速度が遅い

アプリトップ	店舗一覧
1.13s	2.20s

chocoZAPが抱えていた課題

01 | リクエストレスポンスの速度が遅い

chocoZAPが抱えていた課題

01 リクエストレスポンスの速度が遅い

02 DDDの実装によるコードの複雑化と理解の困難

chocoZAPが抱えていた課題

```ruby
module V1
  class UsersController < ApplicationController
    def show
      usecase = ::Users::Usecase::UsersShow.new
      res = usecase.exec(@user_id)

      render json: UserSerializer.new(res).serializable_hash
    end
  end
end
```

chocoZAPが抱えていた課題

```ruby
module Users
  module Usecase
    class UsersShow
      def initialize
        @repo = Users::Repository::UserRepository.new
      end

      def exec(user_id)
        @repo.find_by(user_id)
      end
    end
  end
end
```

chocoZAPが抱えていた課題

```ruby
module Users
  module Repository
    class UserRepository
      RESOURCE = "users"
      def initialize
        @cli = Rizap::Be::Client::BackendClient.new(RESOURCE, Users::Entity::User)
      end

      def find_by(user_id)
        @cli.get("/#{user_id}", entity: Users::Entity::User)
      end
    end
  end
end
```

chocoZAPが抱えていた課題

```ruby
module V1
  class UsersController < ApplicationController
    def show
      usecase = ::Users::Usecase::UsersShow.new
      permitted = params.permit(:id)
      res = usecase.exec(permitted[:id])

      json_string = UserSerializer.new(res).serializable_hash
      render json: json_string
    end
  end
end
```

chocoZAPが抱えていた課題

```ruby
module Users
  module Usecase
    class UsersShow
      def initialize
        @repo = Users::Repositories::UserRepository.new
      end

      def exec(id)
        @repo.find(id)
      end
    end
  end
end
```

chocoZAPが抱えていた課題

```ruby
module Users
  module Repositories
    class UserRepository < ::Repository
      private

      def model_name
        User
      end
    end
  end
end
```

chocoZAPが抱えていた課題

```ruby
class Repository
  def find(id)
    model_name.find(id)
  rescue ActiveRecord::RecordNotFound
    raise System::Error::NotFoundError
  end

  def find_by(key, value)
    model_name.find_by(key => value)
  end
end
```

chocoZAPが抱えていた課題

chocoZAPが抱えていた課題

こういうこと！

```ruby
class UsersController < ApplicationController
  def show
    @user = User.find(params[:id])
  end
end
```

chocoZAPが抱えていた課題

02 DDDの実装によるコードの複雑化と理解の困難

新規メンバーのラーニングコストが高い

オンボーディングへの負荷が高い

chocoZAPが抱えていた課題

02 DDDの実装によるコードの複雑化と理解の困難

chocoZAPが抱えていた課題

01 リクエストレスポンスの速度が遅い

02 DDDの実装によるコードの複雑化と理解の困難

chocoZAPが抱えていた課題

01 リクエストレスポンスの速度が遅い

02 DDDの実装によるコードの複雑化と理解の困難

03 ProxyにしかなっていないBFF

chocoZAPが抱えていた課題

03 | ProxyにしかなっていないBFF

Backend For Frontendなのに、Ruby on Rails製でバックエンドの人間が開発していたため、**ただのProxyにしかなっておらず管理コストが負担に。**

chocoZAPが抱えていた課題

03 | ProxyにしかなっていないBFF

適切な運用がされていないため

不要の判断に

Backend For Frontendなのに、Ruby on Rails製
していたため、**ただ**
管理コストが負担に。

chocoZAP

RIZAP TECHNOLOGIES

課題への解決策としての

リファクタリング

リファクタリングの際にしたこと

リファクタリングの際にしたこと

DDDからMVCへ

全バックエンドメンバーで既存実装の理解を行ったのちにコードを置き換える

リファクタリングの際にしたこと

DDDからMVCへ

全バックエンドメンバーで既存実装の理解を行ったのちにコードを置き換える

BFFの廃止

クライアントから直接BEのAPIを叩くようにし、一つのコントローラーにまとめた

リファクタリングの際にしたこと

DDDからMVCへ

全バックエンドメンバーで既存実装の理解を行ったのちにコードを置き換える

BFFの廃止

クライアントから直接BEのAPIを叩くようにし、一つのコントローラーにまとめた

YJITの導入

Rubyのバージョンを3.1から3.3系にアップグレードしてYJITを導入

※リファクタリング作業中にRuby 3.3.1がリリース

リファクタリング結果と

達成して得られた点

リファクタリング結果

リファクタリング結果

01 | リクエストレスポンスの速度が速くなった

リファクタリング結果

01 リクエストレスポンスの速度が速くなった

アプリトップ

1.13s ▸ 505ms

店舗一覧

2.20s ▸ 14.5ms

リファクタリング結果

01 | リクエストレスポンスの速度が速くなった

chocoZAP

RIZAP TECHNOLOGIES

リファクタリング結果

01 | リクエストレスポンスの速度が速くなった

02 | リクエスト数が減少した

chocoZAP

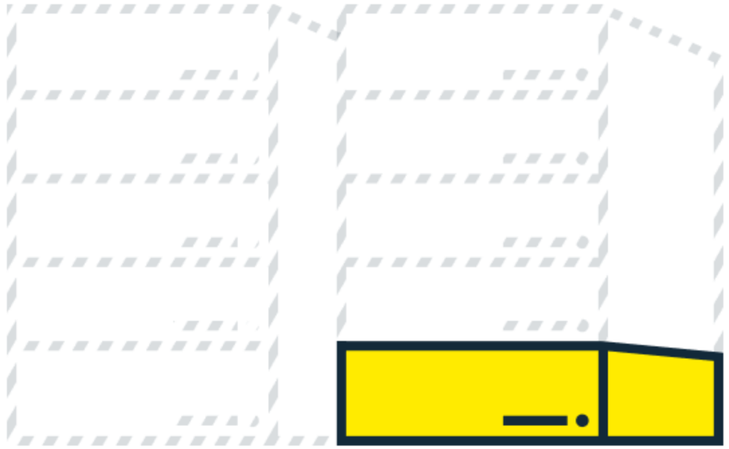
RIZAP TECHNOLOGIES

リファクタリング結果

02 | リクエスト数が減少した

BFFで各コンテンツごとの処理を

1コントローラーに



リファクタリング結果

01 | リクエストレスポンスの速度が速くなった

02 | リクエスト数が減少した

リファクタリング結果

01 | リクエストレスポンスの速度が速くなった

02 | リクエスト数が減少した

03 | AWSインスタンス数が激減しコスト削減に繋がった

リファクタリング結果

03 | AWSインスタンス数が激減しコスト削減に繋がった

三桁の時もあったEC2が

5台以下に

リファクタリング結果

コードがシンプルかつ理解しやすくなった

達成して得られた点

Railsの標準に沿って開発することが一番大事！

標準の道具 × 標準の方法 × 必要十分で作る

Quoted From Kaigi on Rails 2024

chocoZAP

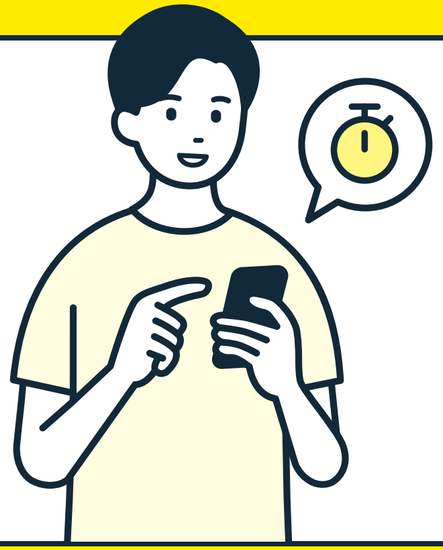
RIZAP TECHNOLOGIES

達成して得られた点

BFFの廃止によって管理コスト減少

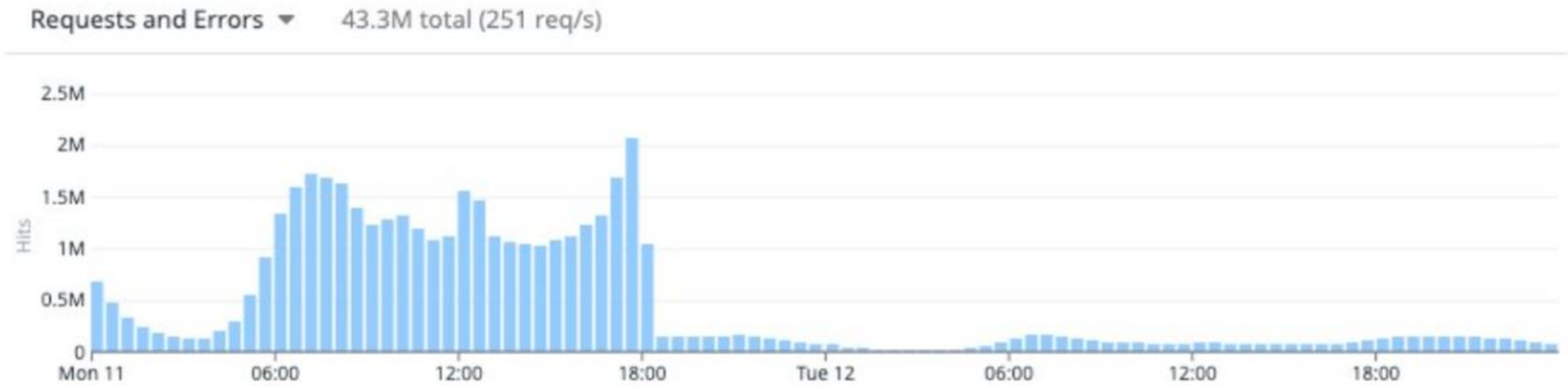
リクエスト数 **約1/100削減**

コントローラーでSQLをまとめたので速度が上がった



余談

BFFで不必要にuser_idの確認をしている処理があり、
該当処理を削除した結果すごいことになった



今後の展望

Rails Wayに沿ったアプリ開発!

chocoZAP

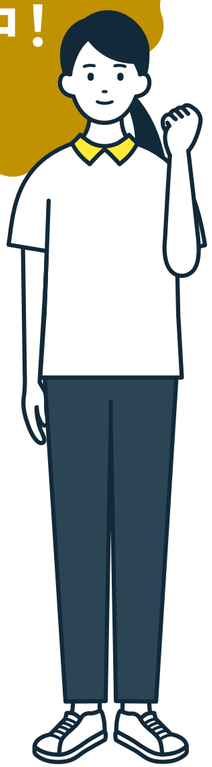
RIZAP TECHNOLOGIES

中途採用について

＼ 新規メンバー募集中！ ／

全ポジション募集中！

Ruby　Swift　Kotlin　SRE　PM　QA　等



noteも見てね

https://note.com/rizap_tech/

RIZAP TECHNOLOGIES

THANK YOU!