

HAL Event Week

SNS APP

A.Yamamoto

目次

- 今回の目標
- 設計
- 開発
 - フローなど
 - CI/CD 準備
 - バックエンド構築
 - フロントエンド構築
 - インフラ構築
- デモ
- 振り返り

目次

- 今回の目標 📌
- 設計
- 開発
 - フローなど
 - CI/CD 準備
 - バックエンド構築
 - フロントエンド構築
 - インフラ構築
- デモ
- 振り返り

今回の目標

- Hello PHP!
- Hello Laravel
- Hello CI/CD
- Modern React
- Stop! 猛コード・ノーテスト開発

今回の目標

完成形



目次

- 今回の目標
- 設計 🖐️
- 開発
 - フローなど
 - CI/CD 準備
 - バックエンド構築
 - フロントエンド構築
 - インフラ構築
- デモ
- 振り返り

設計

要件定義

- Githubへ設置

<u>メールアドレスとパスワードを利用して ユーザ登録ができる</u>	ユーザ情報 認証	High
<u>ユーザ名とメールアドレス、パスワード を変更できる</u>	ユーザ情報 認証	
<u>ユーザ登録後にタイムラインへ画面遷移 できる</u>		
<u>ログイン/ログアウトができる</u>	認証	High
<u>一定時間(2h)放置したら自動でログアウト</u>		
<u>メッセージ投稿ができる</u>	メッセージ	High
<u>いいねができる</u>	メッセージ いいね	High

設計

DB 設計

users				
ユーザID	id	CHAR(36)	PK	バックエンドでUUID生成
ニックネーム	name	VARCHAR(64)	NN	日本語英語問わず64文字上限
メールアドレス	email	VARCHAR(255)	UQ, NN	NN
パスワード	password	VARCHAR(255)	NN	8 文字以上 32 文字以下 ハッシュ化するので変わりうる
登録日時	created_at	TIMESTAMP	NN	JSTで格納する
更新日時	updated_at	TIMESTAMP	NN	JSTで格納する

設計

DB 設計

messages				
メッセージID	id	CHAR(36)	PK	バックエンドでUUID生成
本文	body	TEXT	NN	日本語最大140文字
投稿日時	created_at	TIMESTAMP	NN	JSTで格納する
更新日時	updated_at	TIMESTAMP	NN	JSTで格納する
ユーザID	user_id	CHAR(36)	FK	users.id

設計

DB 設計

favorites				
お気に入りID	id	CHAR(36)	PK	バックエンドでUUID生成
登録日時	created_at	TIMESTAMP	NN	JSTで格納する
更新日時	updated_at	TIMESTAMP	NN	JSTで格納する
ユーザID	user_id	CHAR(36)	FK	users.id
メッセージID	message_id	CHAR(36)	FK	messages.id

設計

URL 設計

OpenAPI スキーマとして吐き出したものをGithubに設置

Swagger UI

画面設計

Figma

Export した SVG をGithubにも置いてあります

目次

- 今回の目標
- 設計
- 開発 🖱️
 - フローなど 🖱️
 - CI/CD 準備
 - バックエンド構築
 - フロントエンド構築
 - インフラ構築
- デモ
- 振り返り

開発

フローなど

- Github Flow を採用
- マージコミットを産んでいいのは PR のマージの時だけ
- マージする前にベースになるブランチは最新にしましょう
 - 最新のベースブランチでリベースしよう


開発





環境

エディタ	IntelliJ IDEA Ultimate / VSCode
バックエンド	Laravel 9 + PHP 8.2.1 (sail 利用)
フロントエンド	React 18 + TypeScript 4.9.4
データベース	MySQL 8.0
インフラ	AWS (IaC 利用、バックエンド) / Cloudflare (バックエンド, フロントエンド)
CI/CD	Github Actions(GitHub-hosted)
認証	Session Auth

開発

フローなど



Update Makefile	20 Jan 2023...	na2na-p	6e9beeb0
Update vite.config.ts	20 Jan 2023...	na2na-p	c7fff546
add tsconfig for tests dir	20 Jan 2023...	na2na-p	3f96ccd2
Add dependencies	20 Jan 2023...	na2na-p	07849048
Move	20 Jan 2023...	na2na-p	16e6394d
 main  origin  origin/HEAD Merge pull request #128 from na2na-p/renovate/all-...	24 Jan 2023...	2na2-p[bot]	4f4c6e9e
remove	24 Jan 2023...	na2na-p	2321af5e
Merge pull request #128 from na2na-p/renovate/all-...	24 Jan 2023...	2na2-p[bot]	2bb2ef72
Update all non-major dependencies	24 Jan 2023...	renovate[bot]	b7762e12
Merge pull request #126 from na2na-p/renovate/vite	23 Jan 2023...	2na2-p[bot]	17e15279
Update vite to ^0.28.1	23 Jan 2023...	renovate[bot]	72692034
 origin/topic/backend/tests/message-list Merge pull request #126 from na2na-p/renovate/vite	23 Jan 2023...	2na2-p[bot]	2e447527
Add SwaggerUI container	23 Jan 2023...	na2na-p	71a5674d

目次

- 今回の目標
- 設計
- 開発 📌
 - フローなど
 - CI/CD 準備 📌
 - バックエンド構築
 - フロントエンド構築
 - インフラ構築
- デモ
- 振り返り

開発

CI/CD 構築

- 今回は Public Repository で開発
 - GitHub-hosted Runner を利用
 - GitGuardian による機密情報漏洩チェック
 - ライブラリのライセンスチェックはなし(主に GPL 系ライセンス)

開発

CI/CD 構築

- ライブラリ等の更新は Renovate に丸投げ
- バックエンドではテストで利用するコマンドを質問して Makefile から利用可能に
- Node.js 環境で使う CI はなんとなく理解してるのでサクッと
- フロントエンド CD は Cloudflare Pages へ。ビルドテストも兼ねています。
 - デプロイ後はキャッシュパーージしましょう
- バックエンド CD は AWS EC2 へ
 - SSM を利用して Run Command することで自動更新

開発


CI/CD 構築

- Branch Protection ルールをよしなに設定
 - 最低 1 名の Approve を必要に
 - 抜け道として Bot に Approve させたりしました
 - 設定の終わったテスト系から順次 Required へ
 - コミットに署名されていることを強制
- 全部の条件を満たしたら Bot が勝手に Merge するように

開発

CI/CD 構築

```
`terraform plan`
```



2na2-p bot commented 11 hours ago • edited ▾

Contributor 😊 ⋮

Backend Infra Plan Result










[CI link](#)

```
No changes. Your infrastructure matches the configuration.
```

開発

CI/CD 構築

PR のたびにこういう光景が広がります。これでも見切れていますが。モノレポの辛いところ

✓		Cloudflare Pages / Publish (pull_request)	Successful in 1m		Details
✓		Enable auto merge / enable-auto-merge (pull_request)	Successful in 6s		Details
✓		Pull Request Labeler / triage (pull_request_target)	Successful in 5s		Details
✓		Terraform Plan / EC2-Plan (pull_request)	Successful in 32s	Required	Details
✓		Test(Backend) / Test-Backend (pull_request)	Successful in 7m	Required	Details
✓		Test(Frontend) / Lint-Frontend (pull_request)	Successful in 1m	Required	Details
⌛		Cloudflare Pages / CachePurge (pull_request)	Skipped		Details
✓		Test(Backend) / Lint-Backend (pull_request)	Successful in 4m	Required	Details
✓		GitGuardian Security Checks	Successful in 1s — No secrets detected ✓	Required	Details

目次

- 今回の目標
- 設計
- 開発 📌
 - フローなど
 - CI/CD 準備
 - バックエンド構築 📌
 - フロントエンド構築
 - インフラ構築
- デモ
- 振り返り

開発

バックエンド構築

- Laravel で API サーバの構築
- CORS の設定周りがかなり雑になってしまった

開発

バックエンド構築

- 初期からある認証用ミドルウェアでは不満があったので、自前で作成したものを適用

```
1  class SessionAuthMiddleware
2  {
3      // PHPDoc省略
4      public function handle(Request $request, Closure $next)
5      {
6          if (is_null($request->user())) {
7              return response([
8                  'message' => 'Unauthorized',
9                  ], 401);
10         }
11         return $next($request);
12     }
13 }
```


開発

バックエンド構築

- ルーティングはまとめてスッキリ書こう(一部抜粋)

```
1  Route::prefix('/v1')->group(function () {
2      Route::middleware('sessionAuth')->group(function () {
3          Route::prefix('/users')->group(function () {
4              Route::controller(UsersController::class)->group(function () {
5                  Route::get('/me', 'findUser');
6                  Route::put('/me', 'updateUser');
7                  Route::put('/me/password', 'updatePassword');
8              });
9          });
10         Route::prefix('/messages')->group(function () {
11             Route::controller(MessagesController::class)->group(function () {
12                 Route::post('/', 'createMessage');
13                 Route::get('/', 'listMessage');
14             });
15             Route::put('/{messageId}/favorite', [FavoriteController::class, 'addFavorite']);
16         });
17     });
18 });
```

開発

バックエンド構築

- Eloquent の `with` 便利
- Eloquent Model の良さを殺さない書き方をしましょう
 - 必要な時以外できる限り `query`, `select` を使用するのは避ける

```
1  $messages = Message::with('favorites')
2      ->withCount(['favorites' => function (Builder $query) use ($userId) {
3          $query->where('user_id', $userId);
4      }])
5      ->with('user')
6      ->when($lastMessageId, function (Builder $query) use ($lastMessageId) {
7          $query->where('id', '<', $lastMessageId);
8      })
9      ->orderBy('id', 'desc')
10     ->take($perPage)
11     ->get();
```

開発

バックエンド構築

- OpenAPI のスキーマ生成を Laravel の実装から行うように
 - `"vyuldashev/laravel-openapi": "^1.8"` を利用しました
 - 出来上がったスキーマをもとにテストが生成できたら嬉しい
 - リクエスト飛ばすためのツールもスキーマから一発で大体の設定が終わって嬉しい
 - ローカルで SwaggerUI 確認しやすくなって嬉しい
 - フロントエンドもスキーマ駆動で開発できて嬉しい

開発

バックエンド構築

- テスト実装
 - DRY 原則に背いたテストコードを書くことも多々ある。むしろ愚直に書くべき。
 - 更新後の値もきちんと確認しよう、更新できて終わりではない
 - テストの意図が伝わりやすいテストを書こう
- カバレッジは確認しておこう
 - 今回は 1 箇所の漏れに気づけた

目次

- 今回の目標
- 設計
- 開発 🖱️
 - フローなど
 - CI/CD 準備
 - バックエンド構築
 - フロントエンド構築 🖱️
 - インフラ構築
- デモ
- 振り返り

開発

フロントエンド構築

- TypeScript で書く
- MUI 利用
- フォームには下記を利用
 - react-hook-form
 - yup

開発

フロントエンド構築

- スキーマ駆動開発で幸せになりましょう
- 型は Orval によって自動生成
 - 自動生成された TanStack Query のクライアントが壊れていて使い物にならなくて悲しい

```
1 export type GetApiV1Messages200Item = {  
2   id: string;  
3   body: string;  
4   created_by: string;  
5   created_at: string;  
6   isFavorite: boolean;  
7   favoritesCount: number;  
8 };
```

開発

フロントエンド構築

- テスタブルな Component/Hooks 実装をしましょう

```
1  // 内部でコンポーネントを呼ぶHook
2  const useHooks = (
3    {
4      handleClick,
5    }: {
6      handleClick: () => void;
7    },
8    HogeComponent: FC<HogeComponentProps>
9  ) => {
10    React.createElement(HogeComponent, {
11      onClick: handleClick,
12    });
13  };
```

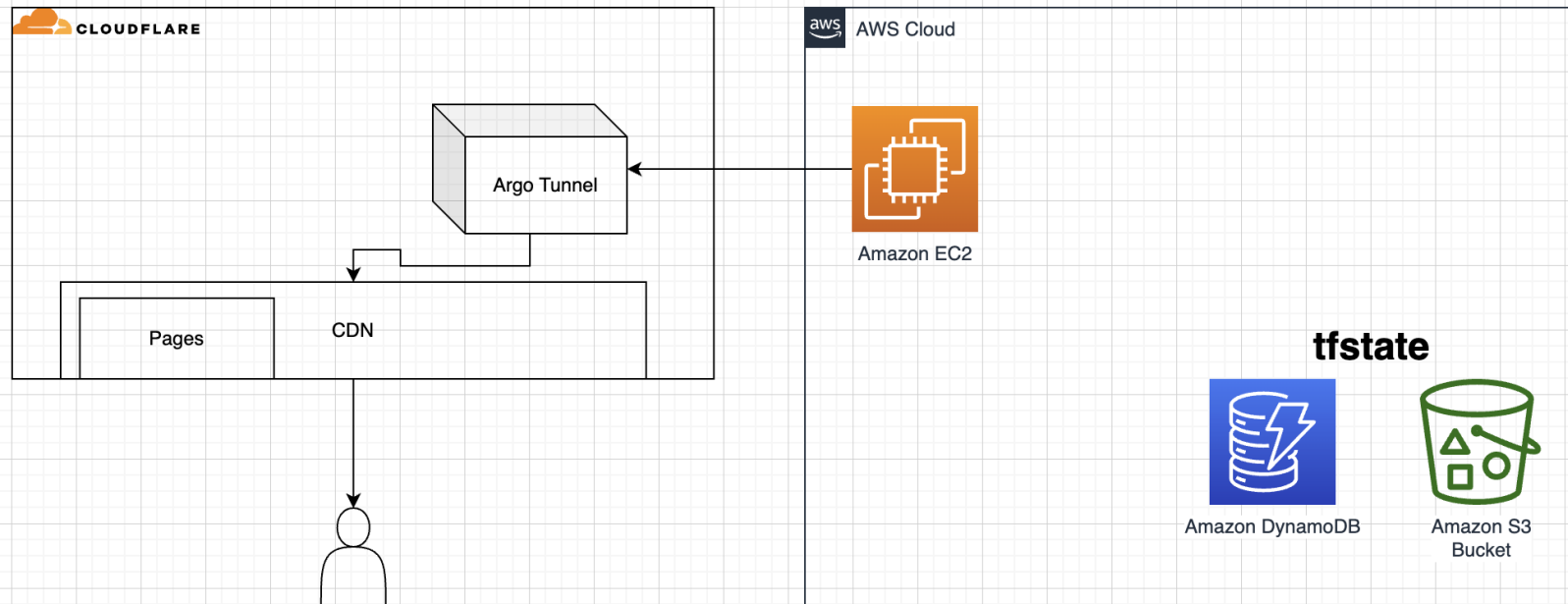

目次

- 今回の目標
- 設計
- 開発 📌
 - フローなど
 - CI/CD 準備
 - バックエンド構築
 - フロントエンド構築
 - インフラ構築 📌
- デモ
- 振り返り

開発

インフラ構築

とてもかんたんな構成図



開発

インフラ構築

AWS EC2 は IaC 利用しました。

tfstate は S3 で管理し、DynamoDB を利用して排他制御をしています。

~~排他制御の影響で main からリベースしたものを矢継ぎ早に force push すると CI 上で動かしている
`terraform plan` が落ちる...~~

開発

インフラ構築

- フロントエンド

Github Actions でビルドして、Cloudflare Pages へデプロイ。

CDN なので、デプロイと同時にキャッシュパージも行う。

[← Pages](#) / sns-app

sns-app

[Deployments](#) [Functions metrics](#) [Custom domains](#) [Settings](#)

Production

[Visit site](#) 

Domains: [sns-app.pages.dev](#), [sns-app.na2na.dev](#)

Production



main

782d6fca.sns-app.pages.dev 

✓ 2 hours ago

[View details](#)

開発

インフラ構築

- バックエンド

EC2 + Cloudflare Tunnel で構築

気持ち的には ECS だったが、イメージ作成が辛そうだったので見送り

sail の使ってるのが php のビルトインサーバだったと思うので、本格的にやるならば避けるべきだと思う

インスタンス (1) 情報

 接続 インスタンスの状態 ▼ アクション ▼ インスタンスを起動 ▼

< 1 > 

インスタンスの状態 = running × フィルターをクリア

<input type="checkbox"/>	Name ▼	インスタンス ID ▼	インスタンス... ▼	インスタンス... ▼	ステータス ▼
<input type="checkbox"/>	na2na-sns-app-backend		 実行中 	t2.micro	 2/2

開発

インフラ構築

- バックエンド
 - Cloudflare Tunnel を使ってることで、ポート解放が不要に

▼ インバウンドルール

🔍 フィルタールール						
名前	セキュリティグループブ...	ポート範囲	プロトコル	ソース	セキュリティグループ	説明
表示するルールがありません						

- 本当なら Public IP も必要ないけれど、SSM 周りがうまくいかなかったので一旦そのままに

開発

インフラ構築

- バックエンド
 - CD 関係では巷でよく見かける、デプロイのタイミングでインバウンドルールに穴を開けるのではなく、SSM で Run Command するようにしています。
 - それ用の Policy / Role を作るのも IaC でやってます。

The screenshot displays the AWS CloudFormation console interface. At the top, a navigation bar shows a back arrow and the text 'Backend CD'. Below this, a green checkmark icon is followed by the resource name 'Add SwaggerUI container #19'. A sidebar on the left contains a 'Summary' section with a home icon, and a 'Jobs' section with a list of tasks: 'Deploy' (highlighted with a green bar and checkmark), 'Run details', 'Usage' (with a clock icon), and 'Workflow file' (with a document icon). The main content area is titled 'Deploy' and indicates the deployment 'succeeded on Jan 23 in 11s'. A search bar labeled 'Search logs' is present. Below the title, a list of deployment steps is shown, each with a chevron icon, a checkmark, and a duration:

Step	Duration
> Set up job	1s
> Checkout	1s
> Configure AWS credentials	1s
> Deploy	4s
> Check Run Command Result	3s
> Post Configure AWS credentials	0s
> Post Checkout	0s

開発

インフラ構築

Cloudflare の WAF もこんな感じで簡単にセットができます

画像が一時期話題だった LOG4J のあれこれの時に仕込んだものです。

簡単すぎるから突破されていそうというのと、CF がデフォルトで対策していそう

Edit firewall rule

Rule name (required)

3 Billion Devices

Give your rule a descriptive name

When incoming requests match...

Field

Operator

Value

URI

contains


`${jndi:ldap`

e.g. `/content?page=1234`

And

Or

目次

- 今回の目標
- 設計
- 開発
 - フローなど
 - CI/CD 準備
 - バックエンド構築
 - フロントエンド構築
 - インフラ構築
- デモ 
- 振り返り

目次

- 今回の目標
- 設計
- 開発
 - フローなど
 - CI/CD 準備
 - バックエンド構築
 - フロントエンド構築
 - インフラ構築
- デモ
- 振り返り 🖐

振り返り

- Laravel バックエンドの構築について
- テスト設計
- REST API の設計について
- インフラ周り
- 総括