具体的な設計

機能洗い出し

画面ベース（UI）

▼ メインページ: `/`

▼ 検索バー

- 検索条件
  1. 距離
     - 距離の下限（length min)
     - 距離の上限（length max）
     - バックエンドは値として持っとく
     - フロントはどちらでも良い
  2. 目安時間 time
     - バックエンドは値として持っとく
     - フロントはどちらでも良い
  3. タグ検索 tags

     ▼ OK例

     - 温泉
     - 食べ物
     - 上級者向け
     - 初心者向け
     - ファミリー
     - ~~トイレあり~~
     - 歴史
     - カフェ
     - 春におすすめ
     - 夏におすすめ
     - 秋におすすめ
     - 冬におすすめ

     ▼ NG例（タグとして採用しない）

     - スポーツ: このサービスの仕様上全て該当してしまうから意味がない
  4. 検索オプション option
     - AND検索/OR検索
     - NOT検索もいいかも…？
- 並び替え
  1. 距離
  2. 目安時間
  3. いいね

▼ ルートのカード（機能削ぎ落としもアリ）

- ルートのタイトル

        - ルートのひとこと説明
        - 距離
        - 目安時間
        - タグ
        - いいね（❤）数
        - ルート詳細ページに遷移するボタン
        - 画像: チェックポイントの風景など１枚
        - 更新日
    - ユーザープロフィールに遷移するボタン

▼ ルート詳細ページ: `/route/[id]`

▼ ルートのカード（機能削ぎ落としもアリ）

- ルートのタイトル
- ルートのひとこと説明
- 距離
- 目安時間
- タグ
- いいね（❤）ボタン
- ~~ブックマークボタン~~ いいねボタンに合併
- ルート詳細ページに遷移するボタン
- 画像: チェックポイントの風景など１枚
- 更新日

- チェックポイントの名称リスト
- 更新日
- チェックポイントなどの画像
- 静的マップの埋め込み
- いいね数は動的なのでAPIを使う
    - いいねAPI を叩いて取得
    - いいね押下時もいいねAPIで処理
- ~~獲得スタンプ一覧~~
    - ~~視覚的にスタンプが押されていくUI~~
    - ~~マス目が埋まるイメージ~~
- ~~獲得スタンプ割合~~
    - ~~どのコースを何%まで踏破したか可視化する~~
- ナビゲーションページに遷移するボタン

▼ ナビゲーションページ: `/navigate/[id]`

- 次のチェックポイント名
    - 画面上の方
    - h1 タグ１つ分くらいのバーナー的な
- マップ
    - リアルタイムで位置情報載せる

    ▼ Google Maps の機能を使えるならルート可変

        - コースアウトしたらルートも変わる
    ▼ ムズそうならルートは固定
        - コースアウトしても位置情報ピンがずれるだけ
- チェックポイント通知
    - チェックポイントに言ったら画面内通知
    - スタップ押してる感じがあればいいよね〜
    - ~~バイブレーションほしいよね〜~~
    - 音
    - プッシュ通知的なUI
    - 時間経過で消える + スライド・スワイプで消える
    - チェックインしたら適宜保存
- 設定ボタン的なの
    - ~~バイブレーションテスト~~
    - チェックポイント一覧ポップアップ風
        - 踏破済みチェックポイント
        - 未踏破チェックポイント
    - 権限要求も兼ねたテストボタン
        - 音量
        - GPS取得
            - GPT取得頻度（将来実装）
        - 通知頻度設定（将来実装）
            - 通知の実装方法によっては不要
    - ヘルプページ（将来実装）へのボタン

▼ ユーザープロフィールページ: `/profile`

- 非ログイン時のみ
    - サインアップ（アカウント新規作成）ボタン
    - ログインボタン
- ログイン時のみ
    - ログアウトボタン
    - バッチ
        - ホバーでルートタイトル
    - 総走行距離（できれば実装したいね）
    - ブックマーク一覧
        - ルートのカード（機能削ぎ落としもアリ）
            - ルートのタイトル
            - ルートのひとこと説明
            - 距離
            - 目安時間
            - タグ
            - いいね（♥）ボタン
            - ~~ブックマークボタン~~ いいねボタンに合併

        - ルート詳細ページに遷移するボタン
        - 画像: チェックポイントの風景など１枚
- ▼ ChatGPT による、将来実装（追加機能？）
    - ▼ ヘルプページ: `/help`

      > ハッカソンでは省略されがちなのでアピールポイントになるかも

        - よくある質問
            - GPSがうまく動かない時の対処法など
            - チェックインができない
            - スタンプが付与されない
        - 初回利用時の操作説明
    - ▼ エラーページ（ステータス画面）: `/error/[error_code]`
        - ヘルプページ（将来実装）へのボタン
        - 例
            - 404ページ
            - GPSがオフ/ブラウザ許可がないなどのエラー表示（できる？）
        - by ChatGPT
            - シングルページで実装するなら簡易メッセージでもいいので用意すると親切
            - エラー発生時の画面遷移や文言だけでも簡単に設計しておくと、デモ時・リリース時に助かります
    - ▼ 運営・管理画面: `/admin`
        - **ルート作成/編集ページ**
            - タイトル、説明文、距離など基本情報の入力欄
            - チェックポイント（位置情報・説明文・画像など）のCRUD
            - タグの登録・編集
            - 緯度経度を手動で入力 or マップ上でピンを置いて取得
            - ルートのサムネイル画像のアップロード
        - **イベント情報の登録ページ**
            - 臨時イベントや期間限定の観光情報を追加できる
            - イベントが終了したら自動的に表示が消える or 手動でオフにする
        - **ユーザー管理ページ（簡易版でも）**
            - 不適切な投稿や通報があった場合を考慮
            - ユーザー凍結、利用履歴の確認など（将来的に）

機能ベース（API）

> https://editor-next.swagger.io を使って、GitHubで確定する予定
> 今後は Notion が develop ブランチ的な扱い
> RESTful API を採用
>
> **オプショナルだと明示されていない限り、必須**

- ▼ error: string // エラーメッセージ

- 必須
- 成功時は空文字列

▼ タグ

- 検索可能なタグの取得: `GetTags`
  - Endpoint: `GET /api/tags`
  - リクエスト: なし
  - ▼ レスポンス
    - Response Body
      - tags: string[] // タグの配列
    - Response Headers
      - `Content-Type: application/json`
    - ▼ HTTP Status Code
      - `HTTP/1.1 200 OK`: 成功
      - `HTTP/1.1 500 Internal Server Error`: サーバー内エラー

▼ ルート

- ルート一覧検索（取得）: `SearchRoutes`
  - Endpoint: `POST /api/search`
  - ▼ リクエスト
    - ▼ Request Body
      - distance // 検索距離の構造体（範囲指定）
        - min: float
          - 64bit
          - 下限: 0.0
          - 初期値: 0.0
          - 条件として指定しない: -1.0
        - max: float
          - 64bit
          - 下限: 0.0
          - 初期値: 0.0
          - 条件として指定しない: -1.0
      - time // 検索所要時間の構造体（範囲指定）
        - min: int
          - 下限: 0
          - 初期値: 0
          - 条件として指定しない: -1
        - max: int
          - 下限: 0
          - 初期値: 0
          - 条件として指定しない: -1
      - tags: string[] // 検索に使うタグ名の配列
        - 要素数

        - 上限: 20
        - 下限: 1
      - 要素１つについて
        - 上限: 10文字
        - 下限: 1文字
      - 初期値: （空文字配列、要素数0）
      - 条件として指定しない: 初期値
    - search_option: string[] // タグの検索方法
      - 初期値: “OR”
      - 取りうる値
        - “AND”
        - “OR”
        - ”NOT”
    - sort // 並び替え（キーと順序の）指定構造体
      - key: string
        - 初期値: “likes”
        - 取りうる値
          - "distance"
          - "time"
          - "likes"
          - "update_at"
      - order: string
        - 初期値: “asc”
        - 取りうる値
          - “asc” // 昇順
          - “desc” // 降順
    - limit: int // 取得する最大件数
      - 初期値: 12
      - 下限: 1
      - 上限: 60
  - Request Headers
    - `Content-Type: application/json`

▼ レスポンス

- Response Body
  - hit_count: int // 条件にヒットした件数
  - routes[] // 次の構造体の配列
    - id: string (UUIDv4)// 詳細に移るためのルートのID（UUIDv4）
    - title: string // タイトル
    - description: string // ひとこと説明
    - distance: float // 距離
    - time: int // 目安時間

        - tags: string[] // タグ
        - likes: int // いいね数（一覧画面では静的なのでAPIを使わない）
        - image: string // 画像のURL（詳細の0番目）
        - update_at: string // 更新日 `yyyy/mm/dd`
      - 受け取った値も返す
    - Response Headers
      - `Content-Type: application/json`
    ▼ HTTP Status Code
    - `HTTP/1.1 200 OK`: 成功
    - `HTTP/1.1 400 Bad Request`: 不正なリクエスト（例: 存在しないキー・値の指定が１つ以上含まれるなどのバリデーションで弾かれた場合）
    - `HTTP/1.1 500 Internal Server Error`: サーバー内エラー
- ルート詳細情報の作成: `PostRoute`
  - Endpoint: `POST /api/routes`
  ▼ リクエスト
    ▼ Request Body
    - title: string // タイトル
      - 上限: 20文字
      - 下限: 1文字
    - description: string // ひとこと説明
      - 上限: 60文字
      - 下限: 1文字
    - full_description: string // 説明詳細
      - 上限: 200文字
      - 下限: 1文字
    - distance: float // 距離
      - 64bit
      - 上限: 符号付き浮動小数64bitの上限
      - 下限: 0.0
    - time: int // 目安時間
      - 64bit
      - 上限: 符号付き64bit整数の上限
      - 下限: 0
    - tags: string[] // タグ
      - 要素数
        - 上限: 20
        - 下限: 1
      - 要素１つについて
        - 上限: 10文字
        - 下限: 1文字
    - total_checkpoints: int // チェックポイントの要素数
      - 上限: 20

        - 下限: 1
    - images: string[] // 画像のURL
        - 要素数
            - 上限: 6
            - 下限: 1
        - 要素１つについて
            - 上限: 1023文字
            - 下限: 8文字
        - URLスキーム: `https://`
        - ドメイン
            - `github.com/sushigon-dev/sagaicle-docs`
            - `x162-43-27-150.static.xvps.ne.jp/sushigon-dev/sagaicle-docs`
            - `153.125.129.128/sushigon-dev/sagaicle-docs`
            - その他デプロイサーバー
        - フォーマットと例
            - `https://example.com/images/{UUIDv4}.{png|jpg|jpeg|webp}`
            - `https://github.com/sushigon-dev/images/{UUIDv4}.{png|jpg|jpeg|webp}`
            - `https://x162-43-27-150.static.xvps.ne.jp/images/0e746155-f52e-4502-83f8-ab91e47abf6f.png`
            - `https://github.com/sushigon-dev/sagaicle-docs/images/0e746155-f52e-4502-83f8-ab91e47abf6f.png`
        - パストラバーサル等の脆弱性があるので本当に気をつける
    - map: string // マップのURL
        - 上限: 1023文字
        - 下限: 8文字
        - URLスキーム: `https://`
        - ドメイン: `www.google.com`
        - フォーマットと例
            - `https://www.google.com/maps/embed?pb=めちゃ長い値`
            - `https://www.google.com/maps/embed?pb=!1m34!1m12!1m3!1d53232.399525649176!2d129.8789995119896!3d33.500727456251056!2m3!1f0!2f0!3f0!3m2!1i1024!2i768!4f13.1!4m5Y6f44CB44CSODQ3LTAwMjIg5L2Q6LOA55yM5ZSQ5rSl5biC6Y-h!3m2!1d33.4460754!2d129.994014599999999!5e0!3m2!1sja!2sjp!4v1740241824233!5m2!1sja!2sjp`
        - パストラバーサル等の脆弱性があるので本当に気をつける
- Request Headers
    - `Authorization: Bearer <JWTトークン>`
    - `Content-Type: application/json`

▼ レスポンス

- Response Body
    - route_id: string (UUIDv4)// 生成したルートのUUIDv4
    - update_at: string // 更新日
        - 日付のフォーマット: `yyyy/mm/dd`
        - DBではミリ秒まで持つ
        - 作成時はバックエンドで上書きする
    - 受け取った値も返す

        - Response Headers
            - `Content-Type: application/json`
        - ▼ HTTP Status Code
            - `HTTP/1.1 201 Created`: 作成成功
            - `HTTP/1.1 400 Bad Request`: 不正なリクエスト（例: 不正なリクエスト・バリデーションで弾かれた場合）
            - `HTTP/1.1 401 Unauthorized`: 未認証
            - `HTTP/1.1 500 Internal Server Error`: サーバー内エラー
- ルート詳細情報の取得: `GetRoute`
    - Endpoint: `GET /api/routes/:route_id`
    - ▼ リクエスト
        - Path Parameters
            - `route_id`: ルートのID
            - 詳細は Endpoint に記載
    - ▼ レスポンス
        - Response Body
            - route_id: string (UUIDv4)// ルートのID
            - title: string // タイトル
            - description: string // ひとこと説明
            - full_description: string // 説明詳細
            - distance: float // 距離
            - time: int // 目安時間
            - tags: string[] // タグ
            - total_checkpoints: int // チェックポイントの要素数
            - images: string[] // 画像のUR
            - map: string // マップのURL
            - update_at: string // 更新日
        - Response Headers
            - `Content-Type: application/json`
        - ▼ HTTP Status Code
            - `HTTP/1.1 200 OK`: 成功
            - `HTTP/1.1 400 Bad Request`: 不正なリクエスト（例: UUIDv4 ではないidの指定などのバリデーションで弾かれた場合）
            - `HTTP/1.1 404 Not Found`: リソースが存在しない（例: UUIDv4 ではあるが存在しない場合）
            - `HTTP/1.1 500 Internal Server Error`: サーバー内エラー

▼ いいね

- ルートがいいね済みか判定: `IsLiked`
    - Endpoint: `GET /api/routes/:route_id/like`
    - **認証必須**
    - ▼ リクエスト
        - Path Parameters
            - `route_id`: ルートのID
            - 詳細は Endpoint に記載

        - Request Headers
            - `Authorization: Bearer <JWTトークン>`
    ▼ レスポンス
        - Response Body
            - route_id: string (UUIDv4)// ルートのID
            - is_liked: bool // いいねしているか
            - likes: int // いいね数
        - Response Headers
            - `Content-Type: application/json`
        ▼ HTTP Status Code
            - `HTTP/1.1 200 OK`: 成功
            - `HTTP/1.1 400 Bad Request`: 不正なリクエスト（例: UUIDv4 ではないidの指定などのバリデーションで弾かれ）
            - `HTTP/1.1 401 Unauthorized`: 未認証
            - `HTTP/1.1 404 Not Found`: リソースが存在しない（例: UUIDv4 ではあるが存在しない場合）
            - `HTTP/1.1 500 Internal Server Error`: サーバー内エラー
- いいね追加: `Like`
    - Endpoint: `POST /api/routes/:route_id/like`
    - **認証必須**
    ▼ リクエスト
        - Path Parameters
            - `route_id`: ルートのID
            - 詳細は Endpoint に記載
        - Request Headers
            - `Authorization: Bearer <JWTトークン>`
    ▼ レスポンス
        - Response Body
            - route_id: string (UUIDv4)// ルートのID
            - likes: int // いいね数
        - Response Headers
            - `Content-Type: application/json`
        ▼ HTTP Status Code
            - `HTTP/1.1 200 OK`: 成功
            - `HTTP/1.1 400 Bad Request`: 不正なリクエスト（例: UUIDv4 ではないidの指定などのバリデーションで弾かれた場合）
            - `HTTP/1.1 401 Unauthorized`: 未認証
            - `HTTP/1.1 404 Not Found`: リソースが存在しない（例: UUIDv4 ではあるが存在しない場合）
            - `HTTP/1.1 500 Internal Server Error`: サーバー内エラー
- いいね削除: `Dislike`
    - Endpoint: `DELETE /api/routes/:route_id/like`
    - **認証必須**
    ▼ リクエスト
        - Path Parameters

        - `route_id`: ルートのID
        - 詳細は Endpoint に記載
      - Request Headers
        - `Authorization: Bearer <JWTトークン>`

    ▼ レスポンス

      - Response Body
        - route_id: string (UUIDv4)// ルートのID
        - likes: int // いいね数
      - Response Headers
        - `Content-Type: application/json`

      ▼ HTTP Status Code

        - `HTTP/1.1 200 OK`: 成功
        - `HTTP/1.1 400 Bad Request`: 不正なリクエスト（例: UUIDv4 ではないidの指定などのバリデーションで弾かれた場合）
        - `HTTP/1.1 401 Unauthorized`: 未認証
        - `HTTP/1.1 404 Not Found`: リクエストが存在しない（例: UUIDv4 ではあるが存在しない場合）
        - `HTTP/1.1 500 Internal Server Error`: サーバー内エラー

▼ チェックポイント

- ルートのチェックポイントを取得: `GetCheckpoints`
  - Endpoint: `GET /api/routes/:route_id/checkpoints`
  - **認証はオプショナル**

  ▼ リクエスト

    - Path Parameters
      - `route_id`: ルートのID
      - 詳細は Endpoint に記載
    - Request Headers
      - `Authorization: Bearer <JWTトークン>`

  ▼ レスポンス

    - Response Body
      - route_id: string (UUIDv4)// ルートのID
      - checkpoints[] // 次の構造体の配列
        - name: string // チェックポイントの名前
        - lat: float // 緯度
        - lng: float // 経度
        - is_visited: boolean // チェックイン済みなら true
          - トークンがない場合や不正な場合はエラーではなく全て false で返す
    - Response Headers
      - `Content-Type: application/json`

    ▼ HTTP Status Code

      - `HTTP/1.1 200 OK`: 成功
      - `HTTP/1.1 400 Bad Request`: 不正なリクエスト（例: UUIDv4 ではないidの指定などのバリデーションで弾かれた場合）

- `HTTP/1.1 401 Unauthorized`: 未認証
- `HTTP/1.1 404 Not Found`: リソースが存在しない（例: UUIDv4 ではあるが存在しない場合）
- `HTTP/1.1 500 Internal Server Error`: サーバー内エラー

- チェックイン実行: `Checkin`
  - Endpoint: `POST /api/routes/:route_id/checkpoints/:checkpoint_index/checkin`
  - **認証必須**
  - ▼ リクエスト
    - Path Parameters
      - `route_id`: ルートのID
        - 詳細は Endpoint に記載
      - `checkpoint_index`: チェックポイントの添字
        - 零始まり
        - 詳細は Endpoint に記載
    - Request Headers
      - `Authorization: Bearer <JWTトークン>`
  - ▼ レスポンス
    - Response Body
      - route_id: // ルートのID
      - checkpoint_index: // チェックポイントの添字
      - visited_count: // 処理後のチェックイン状況
      - total_checkpoints: // チェックポイントの要素数
    - Response Headers
      - `Content-Type: application/json`
    - ▼ HTTP Status Code
      - `HTTP/1.1 200 OK`: 成功
      - `HTTP/1.1 400 Bad Request`: 不正なリクエスト（例: UUIDv4 ではないidの指定などのバリデーションで弾かれた場合）
      - `HTTP/1.1 401 Unauthorized`: 未認証
      - `HTTP/1.1 404 Not Found`: リソースが存在しない（例: UUIDv4 ではあるがそのIDをもつルート存在しない場合）
      - `HTTP/1.1 500 Internal Server Error`: サーバー内エラー

▼ ユーザー情報（バッジ含む）

- ユーザープロフィール取得: `GetProfile`
  - Endpoint: `GET /api/user`
  - **認証必須**
  - ▼ リクエスト
    - Request Headers
      - `Authorization: Bearer <JWTトークン>`
  - ▼ レスポンス
    - Response Body
      - user_name: string // ユーザー名
      - badged_routes[] // 取得済みバッチ構造体
        - id: string // バッヂのルートID

        - title: string // バッヂのタイトル
      - liked_routes[]: // いいね済みのルート構造体
        - id: string // いいね済みのルートID（UUIDv4）
        - title: string // いいね済みのルートタイトル
      - mileage: float // ユーザーが走行済みの距離
        - 64bit
      - total_distance: float // 走行可能距離の最大値
        - 64bit
    - Response Headers
      - `Content-Type: application/json`
      - `Authorization: Bearer <JWTトークン>`
    - ▼ HTTP Status Code
      - `HTTP/1.1 200 OK`: 成功
      - `HTTP/1.1 400 Bad Request`: 不正なリクエスト（例: UUIDv4 でないパスパラメーターの場合）
      - `HTTP/1.1 401 Unauthorized`: 未認証
      - `HTTP/1.1 404 Not Found`: ユーザーが存在しない（例: UUIDv4 ではあるがそのIDをもつユーザーが存在しない場合）
      - `HTTP/1.1 500 Internal Server Error`: サーバー内エラー

▼ ユーザー情報（バッジ含む）

- ユーザープロフィール取得: `GetProfile`
  - Endpoint: `GET /api/user`
  - **認証必須**
  - ▼ リクエスト
    - Request Headers
      - `Authorization: Bearer <JWTトークン>`
  - ▼ レスポンス
    - Response Body
      - user_name: string // ユーザー名
      - badged_routes[] // 取得済みバッチ構造体
        - id: string // バッヂのルートID
        - title: string // バッヂのタイトル
      - liked_routes[]: // いいね済みのルート構造体
        - id: string // いいね済みのルートID（UUIDv4）
        - title: string // いいね済みのルートタイトル
      - mileage: float // ユーザーが走行済みの距離
        - 64bit
      - total_distance: float // 走行可能距離の最大値
        - 64bit
    - Response Headers
      - `Content-Type: application/json`
      - `Authorization: Bearer <JWTトークン>`
    - ▼ HTTP Status Code

- `HTTP/1.1 200 OK`: 成功
- `HTTP/1.1 400 Bad Request`: 不正なリクエスト（例: UUIDv4 でないパスパラメーターの場合）
- `HTTP/1.1 401 Unauthorized`: 未認証
- `HTTP/1.1 404 Not Found`: ユーザーが存在しない（例: UUIDv4 ではあるがそのIDをもつユーザーが存在しない場合）
- `HTTP/1.1 500 Internal Server Error`: サーバー内エラー

▼ ローカル認証

- ユーザ登録（サインアップ）: `Register`
  - Endpoint: `POST /api/auth/register`

  ▼ リクエスト

  - Request Body
    - user_name: string // 一意のユーザー名
      - 上限: 32文字
      - 下限: 1文字
    - password: string // パスワード
      - 上限: 32文字
      - 下限: 1文字
  - Request Headers
    - `Content-Type: application/json`

  ▼ レスポンス

  - Response Body
    - user_name: string // 一意のユーザー名
  - Response Headers
    - `Content-Type: application/json`
    - `Set-Cookie: token=<JWTトークン>`

  ▼ HTTP Status Code

  - `HTTP/1.1 201 Created`: 成功
  - `HTTP/1.1 400 Bad Request`: 不正なリクエスト（例: 文字数が長すぎるなどのバリデーションで弾かれた場合）
  - `HTTP/1.1 409 Conflict`: 一意の値が衝突（例: ユーザー名がコンフリクトした）
  - `HTTP/1.1 500 Internal Server Error`: サーバー内エラー
- ログイン: `Login`
  - Endpoint: `POST /api/auth/login`

  ▼ リクエスト

  - Request Body
    - user_name: string // 一意のユーザー名
      - 上限: 32文字
      - 下限: 1文字
    - password: string // パスワード
      - 上限: 32文字
      - 下限: 1文字
  - Request Headers
    - `Content-Type: application/json`

▼ レスポンス

- Response Body
  - user_name: string // 一意のユーザー名
- Response Headers
  - `Content-Type: application/json`
  - `Set-Cookie: token=<JWTトークン>`

▼ HTTP Status Code

- `HTTP/1.1 200 OK`: 成功
- `HTTP/1.1 400 Bad Request`: 不正なリクエスト（例: 文字数が長すぎるなどのバリデーションで弾かれた場合）
- `HTTP/1.1 401 Unauthorized`: 未認証（例: パスワード違う）
- `HTTP/1.1 500 Internal Server Error`: サーバー内エラー

- ログアウト（トークンのrevoke）: `Logout`
  - Endpoint: `DELETE /api/auth/logout`
  - **認証必須**

▼ リクエスト

- Request Headers
  - `Authorization: Bearer <JWTトークン>`

▼ レスポンス

- Response Body
  - token: // 削除されたJWTトークン
- Response Headers
  - `Content-Type: application/json`

▼ HTTP Status Code

- `HTTP/1.1 200 OK`: 成功
- `HTTP/1.1 400 Bad Request`: 不正なリクエスト（例: トークンの形式が違う場合）
- `HTTP/1.1 401 Unauthorized`: 未認証
- `HTTP/1.1 500 Internal Server Error`: サーバー内エラー

- ユーザー自身の認証情報取得: `Whoami`
  - Endpoint: `GET /api/auth/me`
  - **認証必須**

▼ リクエスト

- Request Headers
  - `Authorization: Bearer <JWTトークン>`

▼ レスポンス

- Response Body
  - user_id: string (UUIDv4)// ユーザーのID
  - user_name: string // ユーザー名
  - token: // JWTトークン
- Response Headers
  - `Content-Type: application/json`

▼ HTTP Status Code

- `HTTP/1.1 200 OK`: 成功
- `HTTP/1.1 400 Bad Request`: 不正なリクエスト（例: トークンの形式が違う場合）
- `HTTP/1.1 401 Unauthorized`: 未認証（ユーザーが存在しないを含む）
- `HTTP/1.1 500 Internal Server Error`: サーバー内エラー

▼ ~~その他~~

▼ OAuth2 を使った認証（後回し）

▼ OAuth2 でユーザ登録: `RegisterWithOAuth`

- **後日検討**

▼ OAuth2 でログイン: `LoginWithOAuth`

- **後日検討**

- ブックマーク（お気に入り）機能: **廃止、いいね機能に合併**

▼ ~~よねぴ（旧アイデア）~~

▼ ルートの検索

▼ 検索

```
// Request
// フォーマットはかたにょに任せます
{
  // 距離検索(範囲指定)
  // 片方null: 上限/下限なし, 両方null: 距離の条件なし
  "distance": {
    "min": Number | null,
    "max": Number | null
  },
  // 所要時間検索(範囲指定)
  "estimated_time": {
    "min": Number | null,
    "max": Number | null
  },
  // タグ検索
  // タグ名(文字列)の配列
  // 直接タグ名ではなくidの配列で管理する？
  "tags": String[],
  // タグの検索方法
  "search_option": "AND" | "OR",
  // 並び替え(キーと昇順/降順の選択)
  // orderはreverse: Booleanとかで管理してもいいかも？
  "sort_by": {
    "key": "distance" | "time" | "likes" | "update_at",
    "order": "asc" | "desc"
  },
  // 取得する最大件数
  // nullで制限なしにする？それは危険？
  "limit": Number
}
```

```
// Response
// このフォーマットで返してほしいです(by よねぴ)
{
  // 条件にヒットした件数
  "hit_num": Number,
```

```
  // 検索結果(上位N件)
  "routes": [
    {
      "id": String,  // ID
      "title": String,  // タイトル
     "description": String,  // ひとこと説明
     "distance_km": Number,  // 距離
     "estimated_time_m": Number,  // 目安時間
     "tags": String[],  // タグ
     "likes": Number,  // いいね数
     "image_url": String,  // 画像のURL
    },
    {
    // 続く...
    }
  ]
}
```

▼ 検索可能なタグの取得

```
// Request
// フォーマットはかたにょに任せます
{
  // なし
}
```

```
// Responce
// フォーマットはかたにょに任せます
{
  "tags": String[]
}
```

▼ 詳細情報の取得

```
// Request
// フォーマットはかたにょに任せます
{
 "id": String  // ID
}
```

```
// Responce
// このフォーマットで返してほしいです(by よねぴ)
{
  "title": String,  // タイトル
  "description": String,  // ひとこと説明
  "distance": Number,  // 距離
  "time": Number,  // 目安時間
  "tags": String[],  // タグ
  "likes": Number,  // いいね数
  "image_url": String,  // 画像のURL → 詳細の方の画像は複数でもいいかも？
  "map_url": String  // マップのURL
}
```

▼ 認証(最低限のユーザー情報を返す)

```
// Request
{
   // クッキーからトークン
}

// Response
{
   "id": String,
   "name": String,
}
```

▼ 機能名

▼ リクエスト

- Path Parameters
  - `/api/users/:id` など
- Query Parameters
  - `?page=1&limit=10` など
- Request Body
  - JSON
  - フォームデータ（ファイルアップロード）
- Request Headers
  - `Authorization: Bearer <JWTトークン>`
  - `Content-Type: application/json`
- Cookies
  - セッションIDや認証情報

▼ レスポンス

- Response Body
  - JSONデータ
  - 本体

▼ HTTP Status Code

- `HTTP/1.1 200 OK`: リクエストが成功したことを示します
- `HTTP/1.1 201 Created`: リクエストは成功し、その結果新たなリソースが作成されたことを示します
- `HTTP/1.1 204 No Content`: リクエストに対して送信するコンテンツはありませんが、ヘッダーは有用であることを示します
  - **今回は使わない**
  - フロントエンドの都合上エラーメッセージも必須（成功時は空文字列）にしたから
- `HTTP/1.1 400 Bad Request`: クライアントのエラーとみなされるもののために、 サーバーがリクエストを処理できない、あるいは処理しようとしない場合を示します
- `HTTP/1.1 401 Unauthorized`: クライアントはリクエストされたレスポンスを得るためには認証を受けなければなりません（未認証）
- ~~`HTTP/1.1 403 Forbidden`: 認可されていないなどの理由でクライアントにコンテンツのアクセス権がなく、サーバーが適切なレスポンスの返信を拒否していることを示します（認証後に拒否）~~
- `HTTP/1.1 404 Not Found`: サーバーがリクエストされたリソースを発見できないことを示します
- `HTTP/1.1 409 Conflict`: このレスポンスは、リクエストがサーバーの現在の状態と矛盾する場合に送られるでしょう。

    - `HTTP/1.1 500 Internal Server Error`：サーバー側で処理方法がわからない事態が発生したことを示します
- Response Headers
    - `Set-Cookie: sessionId=38afes7a8`
    - `Content-Type: application/json`
    - `Authorization: Bearer <JWTトークン>`
    - `CORS設定`
    - `Location: index.html`
- Cookies
    - サーバーからクライアントに`Set-Cookie`で渡す