[RESTful web services](https://ja.wikipedia.org/wiki/Representational_State_Transfer) は軽量で、スケーラビリティーに優れ、メンテナンスがしやすく、webアプリケーションを作成する際に一般的に使用されるものです。

**ザ・ミッション**

RESTfulなwebサービスの[設計原則](https://restfulapi.net/rest-architectural-constraints/" \t "_blank) を満たす、アカウント認証型APIサーバーを実装・公開してください。

**実装の詳細**

**概要**

* 下記のエンドポイントを実装してください:
  + POST /signup -> ユーザアカウントを作成
  + GET /users/{user\_id} -> ユーザ情報を取得
  + PATCH /users/{user\_id} -> ユーザ情報を更新
  + POST /close -> ユーザアカウントを削除
* レスポンスは全てJSON形式で返してください。
* その他明示していない詳細は  
  「RESTfulなwebサービスの[設計原則](https://restfulapi.net/rest-architectural-constraints/" \t "_blank)」  
  に準ずる実装をしてください。
  + [参考文献（日本語）](https://qiita.com/TakahiRoyte/items/949f4e88caecb02119aa)
* 任意のホスティングサービス(Heroku, AWS等)を使用して、アプリをサーバー上にデプロイしてください。
* アプリをデプロイした後、アプリのURLを[env.yml](https://app.tracks.run/editor/env.yml" \t "_blank)に保存してください。
* テストのためのアカウントとして以下を作成しておいてください:
  + user\_id: "TaroYamada"
  + password: "PaSSwd4TY"
  + nickname: "たろー"
  + comment: "僕は元気です"
* テストのために予約されたアカウントとして、"Test～" で始まる user\_id が未作成の状態にしておいてください。
* trackウェブエディター内で「テストを実行」し、実装が要求を満たしているか確認ください。

**基本仕様**

**POST /signup エンドポイント**

* アカウントを新規作成します。
* 期待するrequest形式: POST /signup
  + Body フィールド:
    - user\_id 必須、6文字以上20文字以内、半角英数字
    - password 必須、8文字以上20文字以内、半角英数字記号（空白と制御コードを除くASCII文字）
    - 例）
    - {
    - "user\_id": "TaroYamada",
    - "password": "PaSSwd4TY",

}

* 期待するresponse形式:
  + 成功:
    - ステータスコード: 200
    - レスポンスボディ:
    - {
    - "message": "Account successfully created",
    - "user": {
    - "user\_id": "TaroYamada",
    - "nickname": "TaroYamada"
    - }

}

* + 失敗（送信されたフィールド値に問題がある場合）:
    - ステータスコード: 400
    - レスポンスボディ:
    - {
    - "message": "Account creation failed",
    - "cause": "required user\_id and password"

}

* + - causeにはフィールド値の問題の内容によって適切なエラー理由を記載してください。
      * 必須項目の存在チェック（required）
      * 値の長さのチェック（length）
      * 文字種のチェック（pattern）
  + 失敗（既に同じuser\_idを持つアカウントが存在している場合）:
    - ステータスコード: 400
    - レスポンスボディ:
    - {
    - "message": "Account creation failed",
    - "cause": "already same user\_id is used"

}

**GET /users/{user\_id} エンドポイント**

* 指定user\_idのユーザ情報を返します。
* 期待するrequest形式: GET /users/TaroYamada
  + Authorization ヘッダー: Basic <base64エンコードされた {user\_id} + ':' + {password}>
* 期待するresponse形式:
  + 正常（パターン１：設定済み）:
    - ステータスコード: 200
    - レスポンスボディ:
    - {
    - "message": "User details by user\_id",
    - "user": {
    - "user\_id": "TaroYamada",
    - "nickname": "たろー",
    - "comment": "僕は元気です"
    - }

}

* 期待するresponse形式:
  + 正常（パターン２：未設定または設定なし ）:
    - ステータスコード: 200
    - レスポンスボディ:
    - {
    - "message": "User details by user\_id",
    - "user": {
    - "user\_id": "TaroYamada",
    - "nickname": "TaroYamada"
    - }

}

* + - nickname は未設定時、 user\_id と同じ値を返してください。
  + 失敗 (指定user\_idのユーザ情報が存在しない場合):
    - ステータスコード: 404
    - レスポンスボディ:

{ "message":"No User found" }

* + 失敗 (Authorizationヘッダでの認証が失敗した場合):
    - ステータスコード: 401
    - レスポンスボディ:

{ "message":"Authentication Faild" }

**PATCH /users/{user\_id} エンドポイント**

* 指定idのユーザ情報を更新し、更新したユーザ情報を返します。
* 期待するrequest形式: PATCH /users/TaroYamada
  + Authorization ヘッダー: Basic <base64エンコードされた {user\_id} + ':' + {password}>
  + Body フィールド:
    - nickname 任意、30文字以内、制御コード以外の任意の文字。
      * 空文字を指定すると初期値（ユーザIDに戻る）
    - comment 任意、100文字以内、制御コード以外の任意の文字。
      * 空文字を指定するとクリアされる
    - これらのうち、どちらか片方は必須。送信されない場合は元の値がそのまま残る。
* 期待するresponse形式:
  + 正常:
    - ステータスコード: 200
    - レスポンスボディ:
    - {
    - "message": "User successfully updated",
    - "recipe": [
    - {
    - "nickname": "たろー",
    - "comment": "僕は元気です"
    - }
    - ]

}

* + 失敗 (指定user\_idのユーザ情報が存在しない場合):
    - ステータスコード: 404
    - レスポンスボディ:

{ "message":"No User found" }

* + 失敗 (nickname と comment が両方とも指定されていない場合):
    - ステータスコード: 400
    - レスポンスボディ:
    - {
    - "message": "User updation failed",
    - "cause": "required nickname or comment"

}

* + 失敗 (user\_id や password を変更しようとしている場合):
    - ステータスコード: 400
    - レスポンスボディ:  
      { "message":"Can not change user\_id and password" }
    - {
    - "message": "User updation failed",
    - "cause": "not updatable user\_id and password"

}

* + 失敗 (Authorizationヘッダでの認証が失敗した場合):
    - ステータスコード: 401
    - レスポンスボディ:

{ "message":"Authentication Faild" }

* + 失敗（認証と異なるIDのユーザを指定した場合）:
    - ステータスコード: 403
    - レスポンスボディ:
    - {
    - "message": "No Permission for Update"

}

**Post /close エンドポイント**

* アカウントを削除します。
* 期待するrequest形式: Post /close
  + Authorization ヘッダー: Basic <base64エンコードされた {user\_id} + ':' + {password}>
* 期待するresponse形式:
  + 成功:
    - ステータスコード: 200
    - レスポンスボディ:

{ "message": "Account and user successfully removed" }

* + 失敗 (Authorizationヘッダでの認証が失敗した場合):
    - ステータスコード: 401
    - レスポンスボディ:

{ "message":"Authentication Faild" }