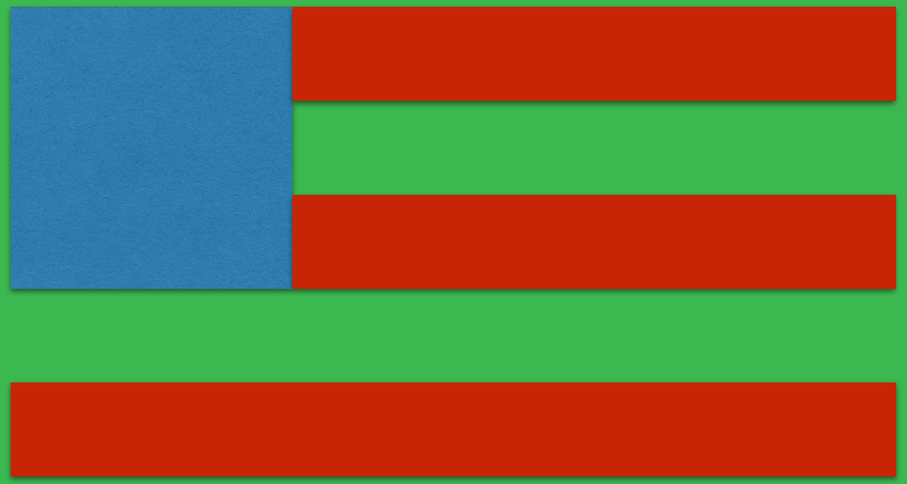
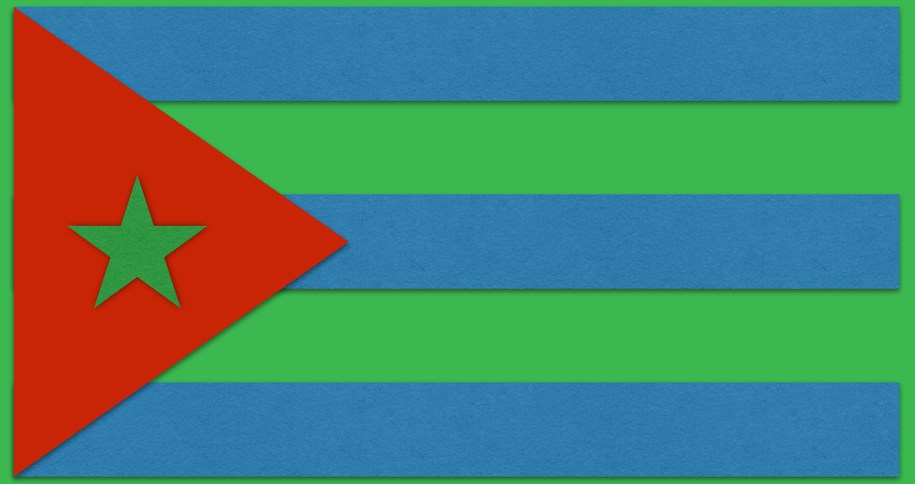
Code With Purpose

Tom Prats

Developer Extraordinaire

@tomprats github.com/tomprats www.tomify.me





traitify

Tomify.me

@tomprats github.com/tomprats www.tomify.me

@tomprats github.com/tomprats www.tomify.me

@tomprats

github.com/tomprats

www.tomify.me

@tomprats

github.com/tomprats

www.tomify.me

Code With Purpose

Why?

How?

What?

Why?

Why do you code?


THE UNITED STATES OF AMERICA
L19745873A

Do you do what you love?

Why I code

He said to them, "Go into all the world and preach the good news to all creation.”

- Mark 16:15

How?

Teach

Do

Open Source

- RapidFTR
- SocialCoding4Good

API Toolbox

@tomprats github.com/tomprats www.tomify.me

traitify

@tomprats github.com/tomprats www.tomify.me

Creating an API

Personify

Your Public Profile

- Single Sign On
- Advertising Profile
- Demographic Search

- Unified Public Profile
- Privacy and Control
- Personal Alerts

Models

Applications
- Public Key
- Private Key
- Approved
- Name

Users
- Email
- Name
- Authentications
- Data

Authentications
- UID
- Token

Data
- Approved
- Origin
- Data

Models

Users

- Email
- Name
- Birthday
- Favorite Color

Models

```
create_table :users do |t|
  t.string :email
  t.string :name
  t.datetime :birthday
  t.string :favorite_color

  t.timestamps
end
```

Authentication

aBcxYz123idk = Base64 Encode “public:secret”

Authorization: Basic aBcxYz123idk

Authentication

http_basic_authenticate_with name: "tomify", password: "secret"

HTTP Basic: Access denied.

Endpoints

/users

Params	Examples
Pagination	?page=3
Limit	?limit=4
	?page=3&limit=4
Search	?name=tom
	?email=tomprats

/users/:id

Params	Examples
Extra Fields	?data=age
	?data=favorite_color

Endpoints

```ruby
def index
  @users = search_users
  @users = @users.page(params[:page]) if params[:page]
  @users = @users.per(params[:limit]) if params[:limit]

  render json: @users, extra_attributes: params[:data]
end

def search_users
  users = User.all

  user_attributes.each do |attr|
    users = users.where("%s ILIKE %s", attr, "$$%#{params[attr]}%$$") if params[attr]
  end

  users
end

def user_attributes
  [:email, :name, :birthday, :favorite_color]
end
```

Endpoints

```
def show
  @user = User.find(params[:id])

  render json: @user, extra_attributes: params[:data]
end
```

Endpoints

curl -u tomify:secret localhost:3000/users?data=age


```
[{
  “id”: 1,
  “email”: “tom@tomprats.com”,
  “name”: “Tom Prats”,
  “birthday”: “1990-11-01T00:00:00.000Z”,
  “age”: 25
}]
```

Consuming an API

Personify Gem

```ruby
require "http"
require "hashie"

class Personify
  def initialize(public_key, private_key, base_url = "")
    @base_url = base_url

    @client = HTTP
      .basic_auth(user: public_key, pass: private_key)
      .headers(accept: :json, content_type: :json)
  end

  def get(path, params = nil)
    objectify @client.get("#{@base_url}#{path}", params: params).parse
  end

  private
  def objectify(hash)
    Hashie::Mash.new hash
  end
end
```

Basics

```
require "http"

class Personify
  def initialize(base_url = "")
    @base_url = base_url

    @client = HTTP
     .headers(accept: :json, content_type: :json)
  end

  def get(path)
   @client.get("#{@base_url}#{path}")
  end
end
```

personify = Personify.new(“http://localhost:3000")
personify.get("/users")

Authentication

```ruby
require "http"

class Personify
  def initialize(public_key, private_key, base_url = "")
    @base_url = base_url

    @client = HTTP
      .basic_auth(user: public_key, pass: private_key)
      .headers(accept: :json, content_type: :json)
  end

  def get(path)
   @client.get("#{@base_url}#{path}")
  end
end
```

```
personify = Personify.new(
  “tomify”, “secret”,
  “http://localhost:3000”
)
personify.get("/users")
```

Parameters

```
require "http"

class Personify
  def initialize(public_key, private_key, base_url = "")
    @base_url = base_url

    @client = HTTP
      .basic_auth(user: public_key, pass: private_key)
      .headers(accept: :json, content_type: :json)
  end

  def get(path, params = nil)
   @client.get("#{@base_url}#{path}", params: params)
  end
end
```

```
personify = Personify.new(
  “tomify”, “secret”,
  “http://localhost:3000”
)
personify.get(“/users”, data: “age”)
```

Parsing

```ruby
require "http"
require "hashie"

class Personify
  def initialize(public_key, private_key, base_url = "")
    @base_url = base_url

    @client = HTTP
      .basic_auth(user: public_key, pass: private_key)
      .headers(accept: :json, content_type: :json)
  end

  def get(path, params = nil)
    objectify @client.get("#{@base_url}#{path}", params: params).parse
  end

  private
  def objectify(hash)
    Hashie::Mash.new hash
  end
end
```

```
personify = Personify.new(
  “tomify”, “secret”,
  “http://localhost:3000”
)
data = personify.get(“/users”, data: “age”)
data.users.first.age
 => 25
```

What?

SupportNate.com

Want to Support Nate?

You probably know that Nate Pettit has recently been injured in a devastating car accident. He's suffered more injuries than I can imagine, but he's fighting through it.

If everyone that has RSVPed *Going* to Prayers for Nate donates just $1.34, then it would take care of $1000 of medical expenses!

To donate money to go towards his medical bills and family then please click below. (Note: Paypal takes 2.9% plus $0.30 per transaction, thus why $1.34 ≈ $1)

Please just remember to keep praying. Nate has a lot left to do in this world and by God's grace we'll see him do it.

Tom’s Missions

missions.tomprats.com

@tomprats github.com/tomprats www.tomify.me

Problem

- No standard
- Not intuitive
- Not customizable
- High cost

Let’s Check It Out!

Models

Users
- Album
- Image
- Email
- Name
- Username
- Admin
- Bio
- Token

Images
- User
- Trip
- Imgur ID
- Link

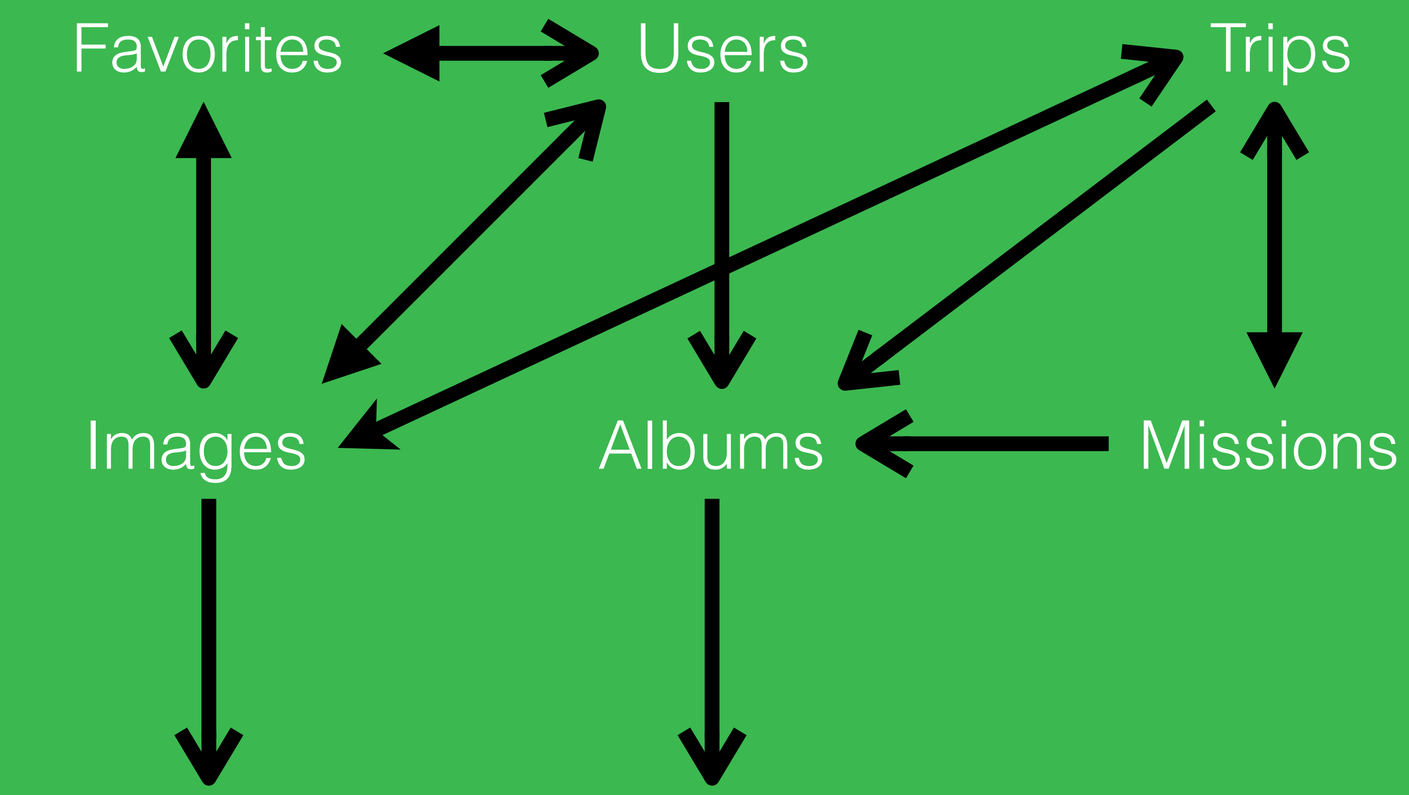
Albums
- Imgur ID

Trips
- Album
- Country
- Start Date
- End Date
- Description

Missions
- User
- Trip
- Album

Favorites
- User
- Image

Models



Many

One

Imgur

Base
- api_get
- api_post
- api_delete
- refresh_token
- token_expired?
- Helper Methods

Album
- all
- get
- create
- add_images
- delete
- Helper Methods

Image
- create
- delete

ImgurError

Imgur

```
module Imgur
  class Base
    def self.api_get(url)
      tries ||= 2
      refresh_token if token_expired?
      headers = { "Authorization" => "Bearer #{ENV["IMGUR_TOKEN"]}" }
      response = Typhoeus.get(url, headers: headers)
      raise ImgurError if response.response_code >= 400
      response
    rescue ImgurError
      error = response.body["data"]["error"]
      refresh_token && (tries -= 1).zero? ? (raise ImgurError.new(error)) : retry
    end
    ...
  end
  ...
end
```

Imgur

```
class Album < Base
  def self.all
    url = "https://api.imgur.com/3/account/#{ENV["IMGUR_USERNAME"]}/albums"
    JSON.parse(api_get(url).body)["data"]
  end
  ...
end
```

Imgur

```
class Image < Base
  def self.create(options)
    url = "https://api.imgur.com/3/image"
    params = {
      image: options[:image].try(:tempfile),
      album: options[:album_id],
      type: options[:type] || "file", # "file" || "base64" || "URL"
      title: options[:title],
      description: options[:description]
    }
    JSON.parse(api_post(url, params).body)["data"]
  end
  ...
end
```

Imgur

```
class Image < ActiveRecord::Base
  ...
  def self.with_imgur(options)
    trip_id = options.delete(:trip_id)
    user_id = options.delete(:user_id)
    image = Imgur::Image.create(options)
    create(
      imgur_id: image["id"],
      link: image["link"],
      trip_id: trip_id,
      user_id: user_id
    )
  end
  ...
end
```

Imgur

```
class MissionsController < ApplicationController
  ...
  def add_images
    image_id = Image.with_imgur(
      title: "#{@mission.trip.name} - #{current_user.username}",
      image: params[:image],
      album_id: @mission.album.imgur_id,
      user_id: current_user.id,
      trip_id: @mission.trip_id
    ).imgur_id

    @mission.trip.album.add_images([image_id])

    render json: { success: true }
  rescue Imgur::ImgurError => e
    logger.debug "Imgur Error #{e.message}"

    render json: { error: e.message }
  end
  ...
end
```

Difficulties

- Design
- Uploads
- Timing

The Future

- Continuous slide show
- Blog functionality
- More user friendly


Just Do It!

Code With Purpose

Questions?

Thank You

God, family, Traitify

@tomprats github.com/tomprats www.tomify.me