

Building an API with Grape

When life gives you ~~lemon~~ grape, make an API

Fabien Loup, September 2022

Real story time

“Client ABC wants an API. They want to be able to do X, Y and Z.”

“YOU HAVE 3 WEEKS”



Grape to the rescue

Grape is a REST-like API framework for Ruby. It's designed to run on Rack or complement existing web application frameworks such as Rails and Sinatra by providing a simple DSL to easily develop RESTful APIs. It has built-in support for common conventions, including multiple formats, subdomain/prefix restriction, content negotiation, versioning and much more.

From the doc: <https://github.com/ruby-grape/grape>



How to start

For example, in a Rails app

```
# Gemfile
gem 'grape'
gem 'grape-entity'
gem 'grape-swagger'
gem 'grape-swagger-rails'
gem 'grape-swagger-entity'
```

```
# config/routes.rb
Rails.application.routes.draw do
  # [...] all your routes

  # We will define this class with grape
  mount API::API => '/'
end
```

How to start

For example, in a Rails app

```
# Gemfile
gem 'grape'
gem 'grape-entity' # wait, what is this?
gem 'grape-swagger'
gem 'grape-swagger-rails'
gem 'grape-swagger-entity'
```

```
# config/routes.rb
Rails.application.routes.draw do
  # [...] all your routes

  # We will define this class with grape
  mount API::API => '/'
end
```

How to start

For example, in a Rails app

```
# Gemfile
gem 'grape'
gem 'grape-entity' # wait, what is this?
gem 'grape-swagger' # swagger-what?
gem 'grape-swagger-rails'
gem 'grape-swagger-entity'
```

```
# config/routes.rb
Rails.application.routes.draw do
  # [...] all your routes

  # We will define this class with grape
  mount API::API => '/'
end
```

How to start

For example, in a Rails app

```
# Gemfile
gem 'grape'
gem 'grape-entity' # wait, what is this?
gem 'grape-swagger' # swagger-what?
gem 'grape-swagger-rails' # why not I guess?
gem 'grape-swagger-entity'
```

```
# config/routes.rb
Rails.application.routes.draw do
  # [...] all your routes

  # We will define this class with grape
  mount API::API => '/'
end
```

How to start

For example, in a Rails app

```
# Gemfile
gem 'grape'
gem 'grape-entity' # wait, what is this?
gem 'grape-swagger' # swagger-what?
gem 'grape-swagger-rails' # why not I guess?
gem 'grape-swagger-entity' # now you are just messing with me
```

```
# config/routes.rb
Rails.application.routes.draw do
  # [...] all your routes

  # We will define this class with grape
  mount API::API => '/'
end
```


The bells and whistles

Provided as-is

```
# app/api/api.rb
module API
  class API < Grape::API

    end
end
```

The bells and whistles

Provided as-is

```
# app/api/api.rb
module API
  class API < Grape::API
    class Root < Grape::API
      helpers ::Authenticate

      rescue_from ActiveRecord::RecordNotFound do |e|
        error!(e.message, 404)
      end

      before { authenticate_user! }

      mount Public::V1::WorkspacesAPI
    end

    format :json
    version 'v1'
    prefix 'api'

    mount Root
    # [...]
  end
end
```

The bells and whistles

Provided as-is

```
# app/api/api.rb
module API
  class API < Grape::API
    class Root < Grape::API
      → helpers ::Authenticate

      → rescue_from ActiveRecord::RecordNotFound do |e|
        error!(e.message, 404)
      end

      → before { authenticate_user! }

      → mount Public::V1::WorkspacesAPI
    end

    → format :json
    → version 'v1'
    → prefix 'api'

    → mount Root
      # [...]
    end
  end
end
```


The bells and whistles

Provided as-is

```
# app/api/api.rb
module API
  class API < Grape::API
    class Root < Grape::API
      → helpers ::Authenticate
      → rescue_from ActiveRecord::RecordNotFound do
        error!(e.message, 404)
      end
      → before { authenticate_user! }
      → mount Public::V1::WorkspacesAPI
    end
    → format :json
    → version 'v1'
    → prefix 'api'
    → mount Root
      # [...]
    end
  end
end
```

```
# app/api/authenticate.rb
module Authenticate
  extend Grape::API::Helpers

  def authenticate_user!
    error!('Access Denied', 401) unless current_user
  end

  def current_user
    @current_user ||= authenticated_user
  end

  def authenticated_user
    # your auth strategy here, returns a User
  end
end
```

The bells and whistles

Provided as-is

```
# app/api/api.rb
module API
  class API < Grape::API
    class Root < Grape::API
      helpers ::Authenticate

      rescue_from ActiveRecord::RecordNotFound do |e|
        error!(e.message, 404)
      end

      before { authenticate_user! }

      mount Public::V1::WorkspacesAPI
    end

    format :json
    version 'v1'
    prefix 'api'

    mount Root
    # [...]
  end
end
```

The bells and whistles

Provided as-is

```
# app/api/api.rb
module API
  class API < Grape::API
    class Root < Grape::API
      helpers ::Authenticate

      rescue_from ActiveRecord::RecordNotFound do |e|
        error!(e.message, 404)
      end

      before { authenticate_user! }

      mount Public::V1::WorkspacesAPI # This is our domain specific API
    end

    format :json
    version 'v1'
    prefix 'api'

    mount Root
    # [...]
  end
end
```


The bells and whistles

Provided as-is

```
# app/api/api.rb
module API
  class API < Grape::API
    class Root < Grape::API
      helpers ::Authenticate

      rescue_from ActiveRecord::RecordNotFound do |e|
        error!(e.message, 404)
      end

      before { authenticate_user! }

      mount Public::V1::WorkspacesAPI # This is our domain specific API
    end

    format :json
    version 'v1'
    prefix 'api'

    mount Root
    # [...]
  end
end
```

But wait

The bells and whistles

Provided as-is

```
# app/api/api.rb
module API
  class API < Grape::API
    class Root < Grape::API
      helpers ::Authenticate

      rescue_from ActiveRecord::RecordNotFound do |e|
        error!(e.message, 404)
      end

      before { authenticate_user! }

      mount Public::V1 # This is our domain specific API
    end

    format :json
    version 'v1'
    prefix 'api'

    mount Root
    # [...]
  end
end
```

But wait

The bells and whistles

Provided as-is

```
# app/api/api.rb
module API
  class API < Grape::API
    class Root < Grape::API
      helpers ::Authenticate

      rescue_from ActiveRecord::RecordNotFound do |e|
        error!(e.message, 404)
      end

      before { authenticate_user! }

      mount Public::V1::WorkspacesAPI # This is our domain specific API
    end

    format :json
    version 'v1'
    prefix 'api'

    mount Root
    # [...]
  end
end
```

But wait

A clever DSL

With spices on top

```
# app/public/v1/workspaces_api.rb
module Public
  module V1
    class WorkspacesAPI < Grape::API
      desc 'Workspaces'
      resource :workspaces do
        desc 'Retrieve all workspaces' do
          tags ['workspaces']
          success Entities::Workspace
          is_array true
        end
        get '/' do
          workspaces = current_user.workspaces
          present workspaces,
            with: Entities::Workspace
        end

        # [...] example with post ----->

        route_param :workspace_id, type: Integer do
          mount Public::V1::Workspaces::DocsAPI
        end
      end
    end
  end
end
```

A clever DSL

With spices on top

```
# app/public/v1/workspaces_api.rb
module Public
  module V1
    class WorkspacesAPI < Grape::API
      desc 'Workspaces'
      resource :workspaces do
        desc 'Retrieve all workspaces' do
          tags ['workspaces']
          success Entities::Workspace
          is_array true
        end
        get '/' do
          workspaces = current_user.workspaces
          present workspaces,
            with: Entities::Workspace
        end

        # [...] example with post ----->

        route_param :workspace_id, type: Integer do
          mount Public::V1::Workspaces::DocsAPI
        end
      end
    end
  end
end
```

```
desc 'Create a workspace' do
  tags ['workspaces']
  success Entities::Workspace
end

params do
  requires :workspace, type: Hash do
    requires :title, type: String
    optional :public, type: Boolean, default: false
  end
end

post '/' do
  workspace = Workspace.new(
    declared(params) [:workspace]
  )

  if workspace.save
    present workspace, with: Entities::Workspace
  else
    error!(
      workspace.errors.full_messages.to_sentence,
      422
    )
  end
end
```

A clever DSL

With spices on top

```
# app/public/v1/workspaces_api.rb
module Public
  module V1
    class WorkspacesAPI < Grape::API
      desc 'Workspaces'
      → resource :workspaces do
        desc 'Retrieve all workspaces' do
          tags ['workspaces']
          success Entities::Workspace
          is_array true
        end
        → get '/' do
          workspaces = current_user.workspaces
          present workspaces,
            with: Entities::Workspace

        end

        # [...] example with post ----->

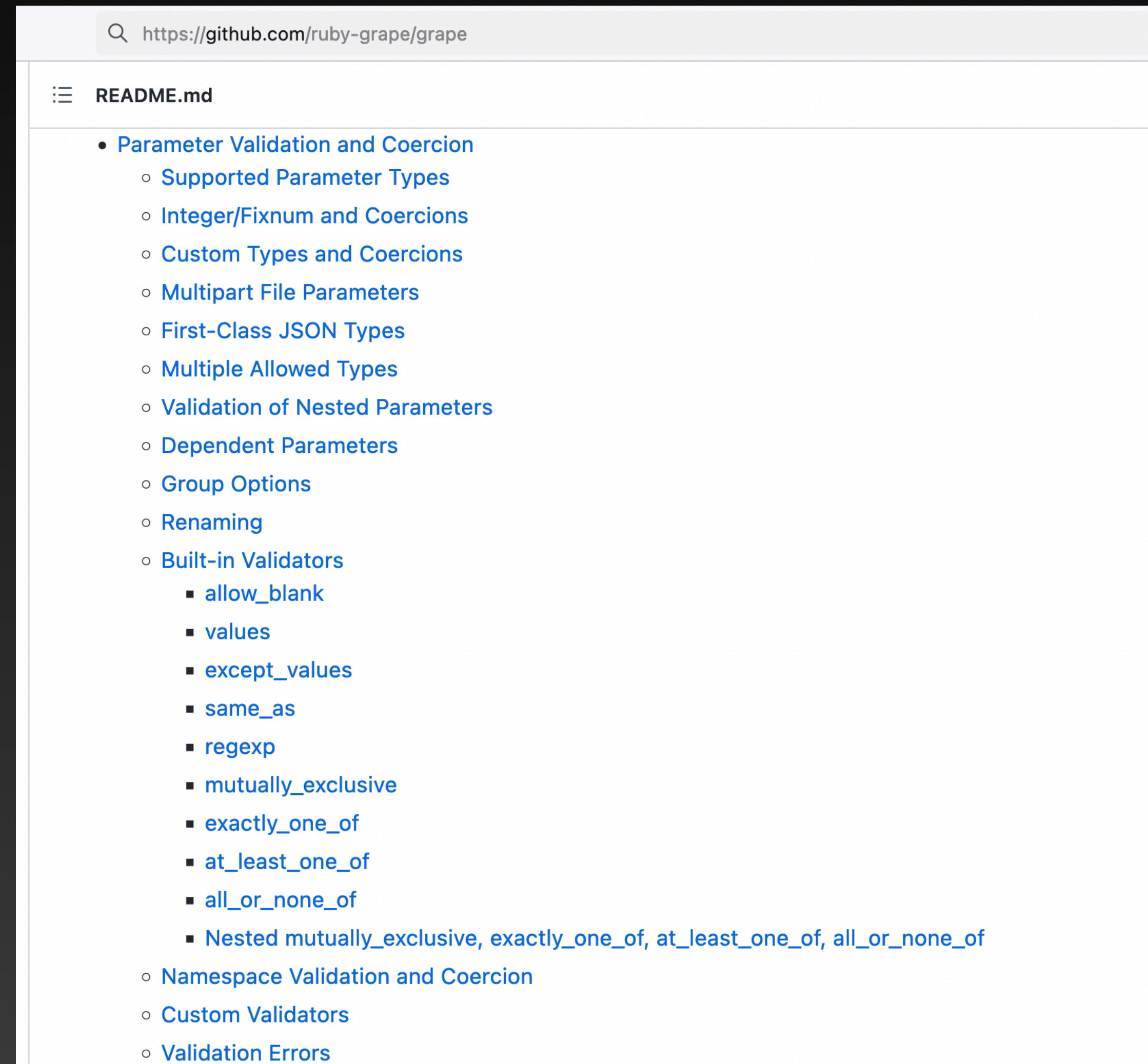
        → route_param :workspace_id, type: Integer do
          mount Public::V1::Workspaces::DocsAPI
        end
      end
    end
  end
end
```

```
desc 'Create a workspace' do
  tags ['workspaces']
  success Entities::Workspace
end
→ params do
  → requires :workspace, type: Hash do
    requires :title, type: String
    → optional :public, type: Boolean, default: false
  end
end
→ post '/' do
  workspace = Workspace.new(
    → declared(params) [:workspace]
  )

  if workspace.save
    present workspace, with: Entities::Workspace
  else
    → error!(
      workspace.errors.full_messages.to_sentence,
      422
    )
  end
end
```


Strong params meets validation

A very extensive API



Q <https://github.com/ruby-grape/grape>

☰ README.md

- Parameter Validation and Coercion
 - Supported Parameter Types
 - Integer/Fixnum and Coercions
 - Custom Types and Coercions
 - Multipart File Parameters
 - First-Class JSON Types
 - Multiple Allowed Types
 - Validation of Nested Parameters
 - Dependent Parameters
 - Group Options
 - Renaming
 - Built-in Validators
 - `allow_blank`
 - `values`
 - `except_values`
 - `same_as`
 - `regexp`
 - `mutually_exclusive`
 - `exactly_one_of`
 - `at_least_one_of`
 - `all_or_none_of`
 - `Nested mutually_exclusive, exactly_one_of, at_least_one_of, all_or_none_of`
 - Namespace Validation and Coercion
 - Custom Validators
 - Validation Errors

A clever DSL

With spices on top

```
# app/public/v1/workspaces_api.rb
module Public
  module V1
    class WorkspacesAPI < Grape::API
      desc 'Workspaces'
      resource :workspaces do
        desc 'Retrieve all workspaces' do
          tags ['workspaces']
          success Entities::Workspace
          is_array true
        end
        get '/' do
          workspaces = current_user.workspaces

          present workspaces, with: Entities::Workspace
        end
      end
    end
  end
end
```

A clever DSL

With spices on top

```
# app/public/v1/workspaces_api.rb
module Public
  module V1
    class WorkspacesAPI < Grape::API
      desc 'Workspaces'
      resource :workspaces do
        desc 'Retrieve all workspaces' do
          tags ['workspaces']
          success Entities::Workspace
          is_array true
        end
        get '/' do
          workspaces = current_user.workspaces
```

```
?? → present workspaces, with: Entities::Workspace
    end
```


A clever DSL

With spices on top

```
# app/public/v1/workspaces_api.rb
module Public
  module V1
    class WorkspacesAPI < Grape::API
      desc 'Workspaces'
      resource :workspaces do
        desc 'Retrieve all workspaces' do
          tags ['workspaces']
          success Entities::Workspace
          is_array true
        end
        get '/' do
          workspaces = current_user.workspaces

          # present interface is pretty cool

          ?? → present workspaces, with: Entities::Workspace
        end
      end
    end
  end
end
```

A clever DSL

With spices on top

```
# app/public/v1/workspaces_api.rb
module Public
  module V1
    class WorkspacesAPI < Grape::API
      desc 'Workspaces'
      resource :workspaces do
        desc 'Retrieve all workspaces' do
          tags ['workspaces']
          success Entities::Workspace
          is_array true
        end
        get '/' do
          workspaces = current_user.workspaces

          # present interface is pretty cool
          # let's use grape-entity
          ?? → present workspaces, with: Entities::Workspace
        end
      end
    end
  end
end
```

A clever DSL

With spices on top

```
# app/public/v1/workspaces_api.rb
module Public
  module V1
    class WorkspacesAPI < Grape::API
      desc 'Workspaces'
      resource :workspaces do
        desc 'Retrieve all workspaces' do
          tags ['workspaces']
          success Entities::Workspace
          is_array true
        end
        get '/' do
          workspaces = current_user.workspaces

          # present interface is pretty cool
          # let's use grape-entity
          ?? → present workspaces, with: Entities::Workspace
        end
      end
    end
  end
end
```

```
# app/api/entities/workspace.rb
module Entities
  class Workspace < Grape::Entity
    expose :id
    expose :title
  end
end
```

A clever DSL

With spices on top

```
# app/public/v1/workspaces_api.rb
module Public
  module V1
    class WorkspacesAPI < Grape::API
      desc 'Workspaces'
      resource :workspaces do

        desc 'Retrieve all workspaces' do
          tags ['workspaces']
          success Entities::Workspace
          is_array true
        end
        get '/' do
          workspaces = current_user.workspaces
          present workspaces, with: Entities::Workspace
        end
      end
    end
  end
end
```


A clever DSL

With spices on top

```
# app/public/v1/workspaces_api.rb
module Public
  module V1
    class WorkspacesAPI < Grape::API
      desc 'Workspaces'
      resource :workspaces do

        desc 'Retrieve all workspaces' do
          tags ['workspaces']
          success Entities::Workspace
          is_array true
        end
        get '/' do
          workspaces = current_user.workspaces
          present workspaces, with: Entities::Workspace
        end
      end
    end
  end
end
```

?? →

A clever DSL

With spices on top

```
# app/public/v1/workspaces_api.rb
module Public
  module V1
    class WorkspacesAPI < Grape::API
      desc 'Workspaces'
      resource :workspaces do
        # using grape-swagger for documentation
        desc 'Retrieve all workspaces' do
          tags ['workspaces']
          success Entities::Workspace
          is_array true
        end
        get '/' do
          workspaces = current_user.workspaces
          present workspaces, with: Entities::Workspace
        end
      end
    end
  end
end
```

?? →

A clever DSL

With spices on top

```
# app/public/v1/workspaces_api.rb
module Public
  module V1
    class WorkspacesAPI < Grape::API
      desc 'Workspaces'
      resource :workspaces do
        # using grape-swagger for documentation
        desc 'Retrieve all workspaces' do
          tags ['workspaces']
          success Entities::Workspace
          is_array true
        end
        get '/' do
          workspaces = current_user.workspaces
          present workspaces, with: Entities::Workspace
        end

        params do
          requires :workspace, type: Hash do
            requires :title, type: String, documentation: { desc: 'Workspace title' }
            optional :public, type: Boolean, default: false
          end
        end
      end
      post '/' do
```

?? →

A clever DSL

With spices on top

```
# app/public/v1/workspaces_api.rb
module Public
  module V1
    class WorkspacesAPI < Grape::API
      desc 'Workspaces'
      resource :workspaces do
        # using grape-swagger for documentation
        desc 'Retrieve all workspaces' do
          tags ['workspaces']
          success Entities::Workspace
          is_array true
        end
        get '/' do
          workspaces = current_user.workspaces
          present workspaces, with: Entities::Workspace
        end

        params do
          requires :workspace, type: Hash do
            requires :title, type: String, documentation: { desc: 'Workspace title' }
            optional :public, type: Boolean, default: false
          end
        end
      end
    end
  end
end
post '/' do
```

?? →

```
# with grape-swagger-entity
module Entities
  class Workspace < Grape::Entity
    expose :id, documentation: {
      type: Integer,
      example: 1024,
      required: true,
    }
    expose :title, documentation: {
      type: String,
      example: 'My place',
      required: true,
    }
  end
end
```

→

More documentation examples

Root doc

```
# app/api/api.rb
module API
  class API < Grape::API
    class Root < Grape::API
      helpers ::Authenticate

      rescue_from ActiveRecord::RecordNotFound do |e|
        error!(e.message, 404)
      end

      before { authenticate_user! }

      mount Public::V1::WorkspacesAPI
    end

    format :json
    version 'v1'
    prefix 'api'

    mount Root
    # [...]
  end
end
```

More documentation examples

Root doc

```
# app/api/api.rb
module API
  class API < Grape::API
    class Root < Grape::API
      end

      format :json
      version 'v1'
      prefix 'api'

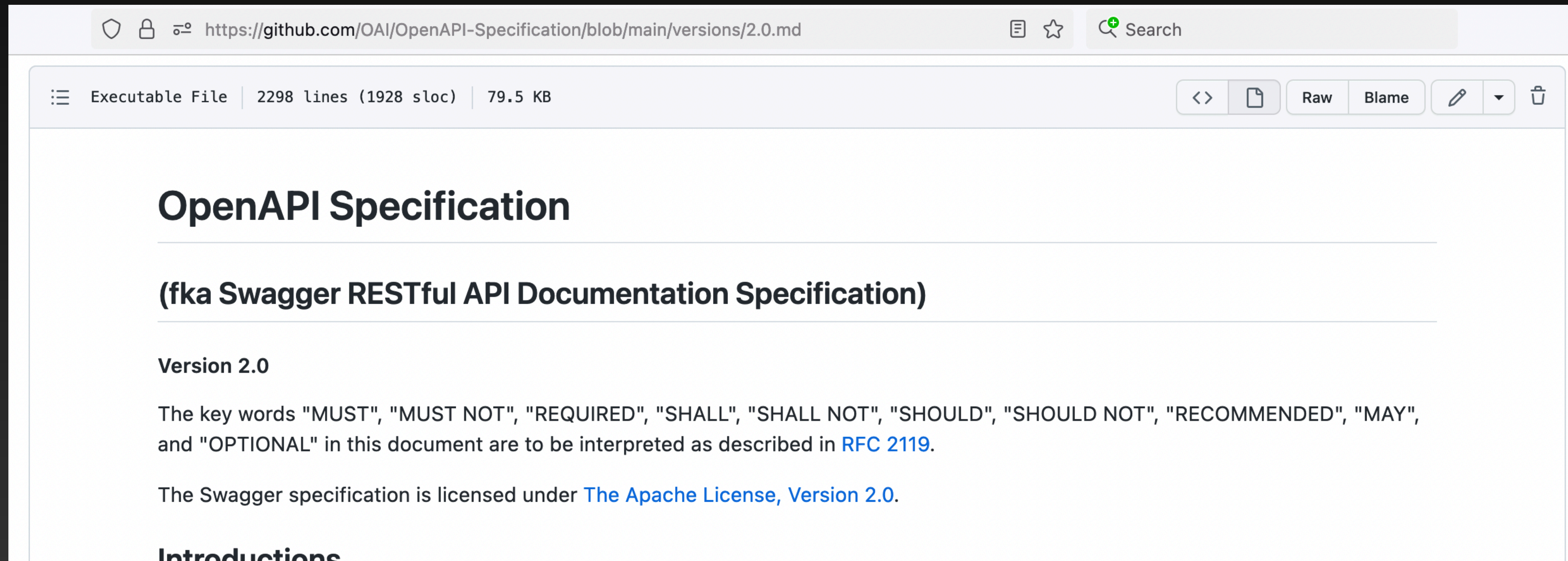
      add_swagger_documentation json: true,
        doc_version: '1.0.0',
        api_version: 'v1',
        mount_path: '/open_api_doc',
        info: {
          title: 'DocSys API',
          description: <<~TEXT,
            If you use _markdown_ *here*, `grape-swagger-rails` will support it,

            ## For ex titles and even

            | GitHub | flavoured | like | tables |
            |-----|-----|-----|-----|
            | Pretty | cool      | no?  | yeah   |
          >>
        }
    end
  end
end
```

Documenting all the things

Swagger / OpenAPI Spec 2.0.



The screenshot shows a web browser displaying the OpenAPI Specification 2.0 document on GitHub. The browser's address bar shows the URL `https://github.com/OAI/OpenAPI-Specification/blob/main/versions/2.0.md`. The GitHub interface includes a search bar and navigation icons. The document header indicates it is an "Executable File" with "2298 lines (1928 sloc)" and a size of "79.5 KB". The document content begins with the title "OpenAPI Specification", followed by the subtitle "(fka Swagger RESTful API Documentation Specification)". Below this, the section "Version 2.0" is introduced. The text explains that key words like "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" are interpreted as described in RFC 2119. It also states that the Swagger specification is licensed under The Apache License, Version 2.0. The section "Introductions" is partially visible at the bottom.

https://github.com/OAI/OpenAPI-Specification/blob/main/versions/2.0.md

Search

Executable File | 2298 lines (1928 sloc) | 79.5 KB

<> [icon] Raw Blame [icon] [icon]

OpenAPI Specification

(fka Swagger RESTful API Documentation Specification)

Version 2.0

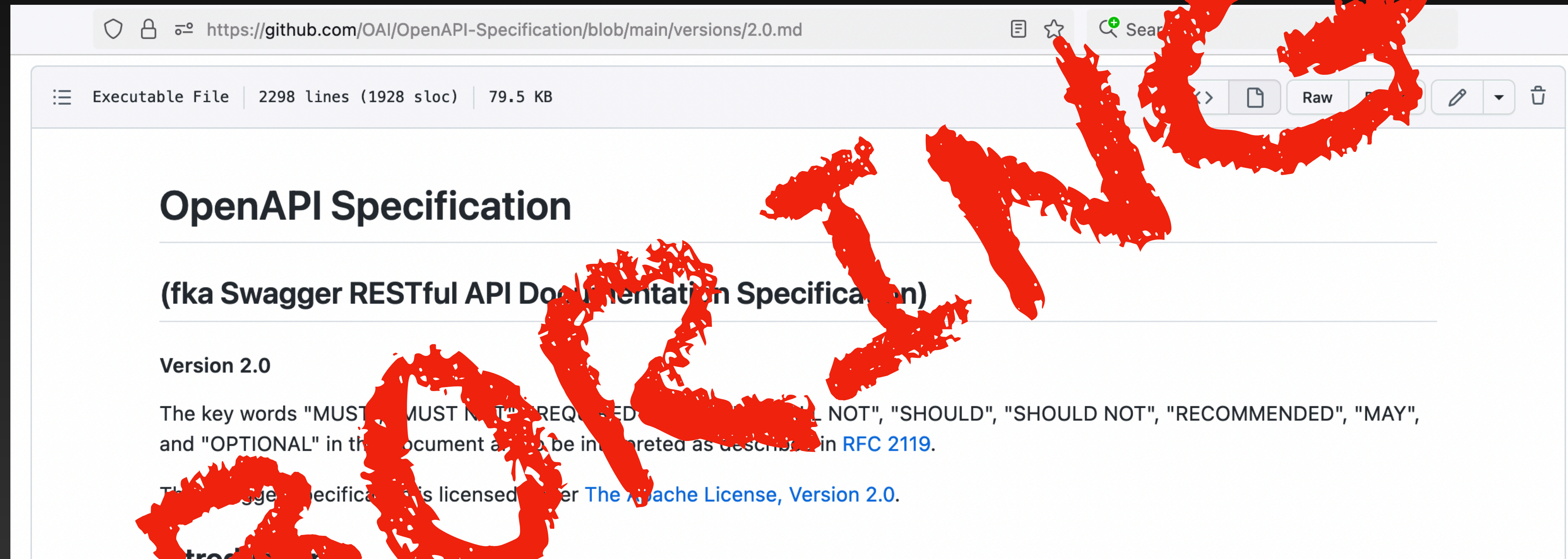
The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC 2119](#).

The Swagger specification is licensed under [The Apache License, Version 2.0](#).

Introductions

Documenting all the things

Swagger / OpenAPI Spec 2.0.



The screenshot shows a web browser displaying the OpenAPI Specification 2.0 document on GitHub. The URL in the address bar is <https://github.com/OAI/OpenAPI-Specification/blob/main/versions/2.0.md>. The document is titled "OpenAPI Specification" and is noted as "(fka Swagger RESTful API Documentation Specification)". It is Version 2.0. The text indicates that key words like "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" are interpreted as described in RFC 2119. It also states that the specification is licensed under The Apache License, Version 2.0. A large, stylized red "WARNING" watermark is overlaid diagonally across the entire page.

https://github.com/OAI/OpenAPI-Specification/blob/main/versions/2.0.md

Executable File | 2298 lines (1928 sloc) | 79.5 KB

OpenAPI Specification

(fka Swagger RESTful API Documentation Specification)

Version 2.0

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC 2119](#).

This specification is licensed under [The Apache License, Version 2.0](#).

More documentation examples

Root doc

```
# app/api/api.rb
module API
  class API < Grape::API
    class Root < Grape::API
      end

      format :json
      version 'v1'
      prefix 'api'

      add_swagger_documentation json: true,
                                doc_version: '1.0.0',
                                api_version: 'v1',
                                mount_path: '/open_api_doc',
                                info: {
                                  title: 'DocSys API',
                                  description: <<~TEXT,
                                    If you use _markdown_ *here*, `grape-swagger-rails` will support it,

                                    ## For ex titles and even

                                    | GitHub | flavoured | like | tables |
                                    |-----|-----|-----|-----|
                                    | Pretty | cool      | no?  | yeah   |
                                }

```

More documentation examples

Root doc

```
# app/api/api.rb
module API
  class API < Grape::API
    class Root < Grape::API
      end

      format :json
      version 'v1'
      prefix 'api'

      add_swagger_documentation json: true,
        doc_version: '1.0.0',
        api_version: 'v1',
        mount_path: '/open_api_doc', # Let's open this url /api/v1/open_api_doc
        info: {
          title: 'DocSys API',
          description: <<~TEXT,
            If you use _markdown_ *here*, `grape-swagger-rails` will support it,

            ## For ex titles and even

            | GitHub | flavoured | like | tables |
            |-----|-----|-----|-----|
            | Pretty | cool      | no?  | yeah   |
          >>
        }
    end
  end
end
```


Document all the things

Swagger / OpenAPI Spec 2.0.

```
localhost:3000/api/v1/open_api_doc

{
  info: {
    title: "Almanac API",
    description: "This is the documentation for Almanac API V1 (Beta access).\nIt is presented in the OpenAPI standard, v2.0.\nYou can consume the schema here:\n<a href=\"/api/v1/open_api_doc\">/api/v1/open_api_doc</a>.\n\n## Authorization\n\nTo request access to the API, please reach out to\n<a href=\"mailto:api@almanac.io\">api@almanac.io</a> with the below information:\n\n- Account email\n- Workspace name\n- Intended use for the API\n\nWe will then provide you with an API access token for your user account.\nIf you have multiple users at your company intending to use the API, please submit\nindividual requests for access tokens as they are **per user**.\n\nTo authenticate to the API, pass the provided access token as an `Authorization`\nrequest header in the form of `Authorization: 'Bearer your-api-key-here'`.\n\nAccess tokens mirror the file system permission of the user they were assigned to.\nFor instance, if user `Pam Beesly` has write permissions to a folder `Office Policies`\nand was given the access token `123XyZ`, requests to the API authenticated with access\ntoken `123XyZ` would have write permissions to the folder `Office Policies`.\n\n## Rate limiting\n\nThe API is rate limited to 10 requests per 10 seconds per access token, using the\nFixed window algorithm.\n\nWhen the rate limit is reached, the request response will have status code `429`\nand body `\"Retry later\\n\\n\"`.\n\n## Errors\n\nWhen an error occurs, the request response will have a specific status code and a\npayload of this format:\n\n`{ error: \"String error message\" }`\n\nThe possible errors are:\n\n| Code | Definition | Description |\n|-----|-----|-----|\n| 400 | Bad Request | Most likely an error in the request parameters (missing parameter, wrong type, etc). Error message to give more information. |\n| 401 | Unauthorized | The access token is invalid. |\n| 403 | Forbidden | The user owning the (valid) access token does not have the permission level to proceed. |\n| 404 | Not Found | The resource is not found. |\n| 429 | Too Many Requests | The rate limit has been hit for this access token. |\n| 422 | Unprocessable Entity | The request parameters are in the correct format but there is a semantic error. Error message to give more information. |\n\n",
    version: "1.0.0"
  },
  swagger: "2.0",
  produces: [
    "application/json"
  ],
  host: "localhost:3000",
  tags: [...],
  paths: {...},
  definitions: {...}
}
```


Document all the things

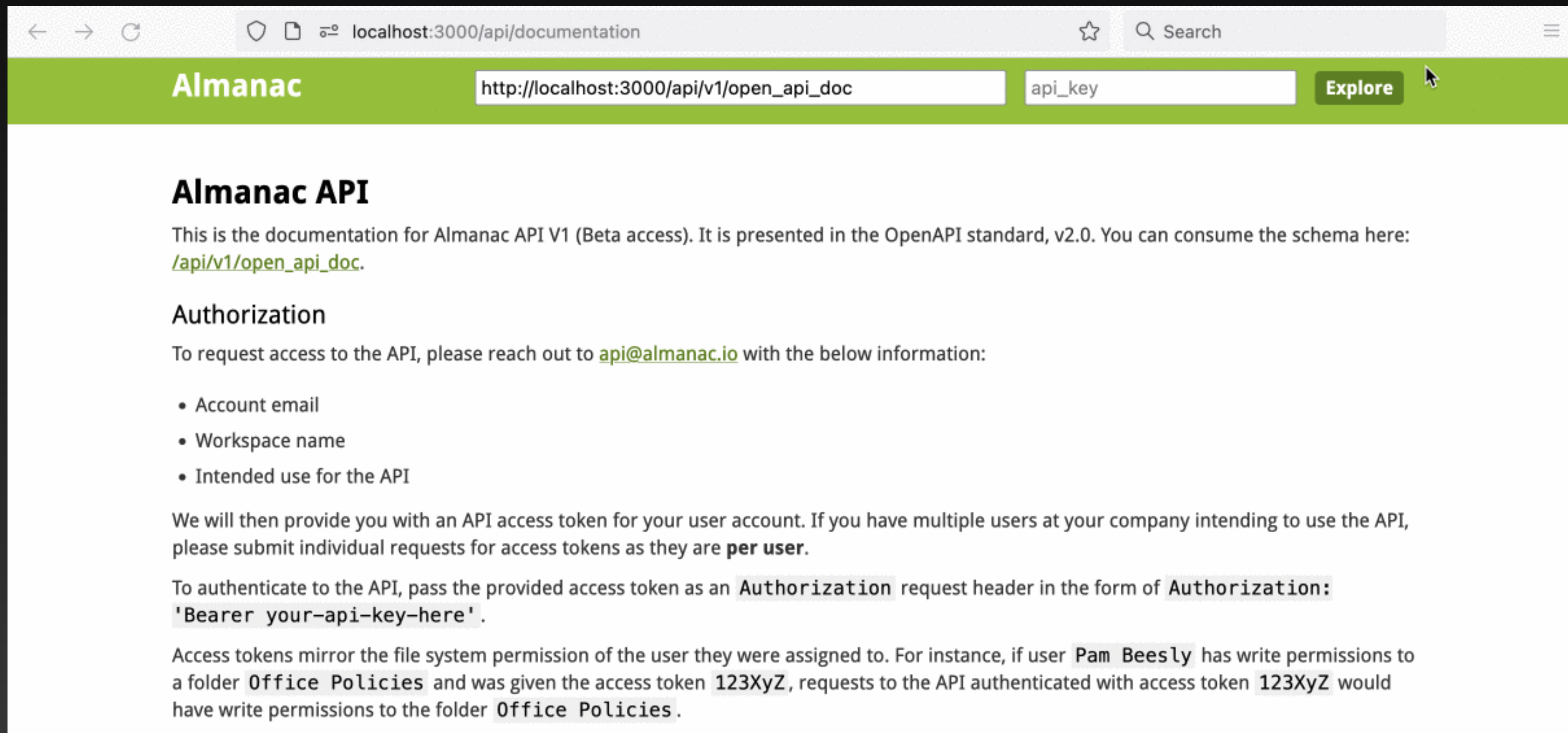
Swagger / OpenAPI Spec 2.0.

```
localhost:3000/api/v1/open_api_doc

{
  info: {
    title: "Almanac API",
    description: "This is the documentation for Almanac API V1 (Beta access).\nIt is presented in the OpenAPI standard, v2.0.\nYou can consume the schema here:\n<a href=\"/api/v1/open_api_doc\">/api/v1/open_api_doc</a>.\n\n## Authorization\n\nTo request access to the API, please reach out to\n<a href=\"mailto:api@almanac.io\">api@almanac.io</a> with the below information:\n\n- Account email\n- Workspace name\n- Intended use for the API\n\nWe will then provide you with an API access token for your user account.\nIf you have multiple users at your company intending to use the API, please submit\nindividual requests for access tokens as they are **per user**.\n\nTo authenticate to the API, pass the provided access token as an `Authorization`\nrequest header in the form of `Authorization: 'Bearer your-api-key-here'`.\n\nAccess tokens mirror the file system permission of the user they were assigned to.\nFor instance, if user `Pam Beesly` has write permissions to a folder `Office Policies`\nand was given the access token `123XyZ`, requests to the API authenticated with access\ntoken `123XyZ` would have write permissions to the folder `Office Policies`.\n\n## Rate limiting\n\nThe API is rate limited to 10 requests per 10 seconds per access token, using the\nFixed window algorithm.\n\nWhen the rate limit is reached, the request response will have status code `429`\nand body `\"Retry later\\n\\n\"`.\n\n## Errors\n\nWhen an error occurs, the request response will have a specific status code and a\npayload of this format:\n\n`{ error: \"String error message\" }`\n\nThe possible errors are:\n\n| Code | Definition | Description |\n|-----|-----|-----|\n| 400 | Bad Request | Most likely an error in the request parameters (missing parameter, wrong type, etc). Error message to give more information. |\n| 401 | Unauthorized | The access token is invalid. |\n| 403 | Forbidden | The user owning the (valid) access token does not have the permission level to proceed. |\n| 404 | Not Found | The resource is not found. |\n| 429 | Too Many Requests | The rate limit has been hit for this access token. |\n| 422 | Unprocessable Entity | The request parameters are in the correct format but there is a semantic error. Error message to give more information. |\n\n",
    version: "1.0.0"
  },
  swagger: "2.0",
  produces: [
    "application/json"
  ],
  host: "localhost:3000",
  tags: [...],
  paths: {...},
  definitions: {...}
}
```


Document all the things

If you are lazy like me, grape-swagger-rails



The screenshot shows a web browser window with the URL `localhost:3000/api/documentation`. The page has a green header bar with the word "Almanac" on the left. In the center of the header is a text input field containing `http://localhost:3000/api/v1/open_api_doc`. To the right of this field is another text input field labeled "api_key". Further right is a green button labeled "Explore".

Almanac API

This is the documentation for Almanac API V1 (Beta access). It is presented in the OpenAPI standard, v2.0. You can consume the schema here: /api/v1/open_api_doc.

Authorization

To request access to the API, please reach out to api@almanac.io with the below information:

- Account email
- Workspace name
- Intended use for the API

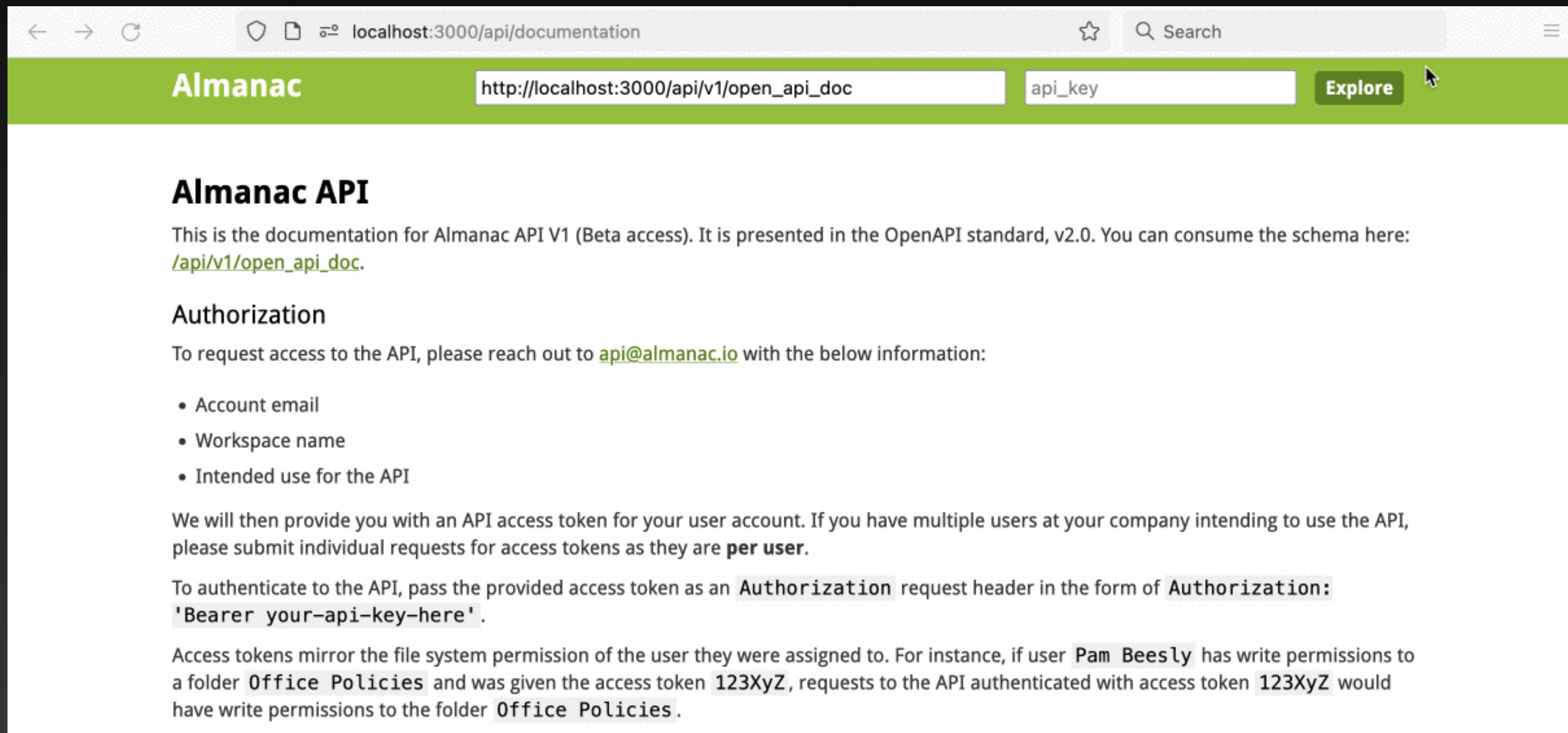
We will then provide you with an API access token for your user account. If you have multiple users at your company intending to use the API, please submit individual requests for access tokens as they are **per user**.

To authenticate to the API, pass the provided access token as an **Authorization** request header in the form of **Authorization:** **'Bearer your-api-key-here'**.

Access tokens mirror the file system permission of the user they were assigned to. For instance, if user **Pam Beesly** has write permissions to a folder **Office Policies** and was given the access token **123XyZ**, requests to the API authenticated with access token **123XyZ** would have write permissions to the folder **Office Policies**.

Document all the things

If you are lazy like me, grape-swagger-rails



The screenshot shows a web browser window with the address bar displaying `localhost:3000/api/documentation`. The page has a green header bar with the word "Almanac" on the left. In the center of the header is a text input field containing `http://localhost:3000/api/v1/open_api_doc`. To the right of this field is another input field labeled "api_key". Further right is a green button labeled "Explore".

Almanac API

This is the documentation for Almanac API V1 (Beta access). It is presented in the OpenAPI standard, v2.0. You can consume the schema here: /api/v1/open_api_doc.

Authorization

To request access to the API, please reach out to api@almanac.io with the below information:

- Account email
- Workspace name
- Intended use for the API

We will then provide you with an API access token for your user account. If you have multiple users at your company intending to use the API, please submit individual requests for access tokens as they are **per user**.

To authenticate to the API, pass the provided access token as an **Authorization** request header in the form of **Authorization:** `'Bearer your-api-key-here'`.

Access tokens mirror the file system permission of the user they were assigned to. For instance, if user **Pam Beesly** has write permissions to a folder **Office Policies** and was given the access token **123XyZ**, requests to the API authenticated with access token **123XyZ** would have write permissions to the folder **Office Policies**.

Real story time

“Client ABC wants an API. They want to be able to do X, Y and Z.”

“YOU HAVE 3 WEEKS”

Did we make it?



Real story time

“Client ABC wants an API. They want to be able to do X, Y and Z.”

“YOU HAVE 3 WEEKS”

Did we make it?



No

No

But it was not grape's fault