Ruby on Rails API-Only Project Structure

Rails in API mode—no views, no asset pipeline. Optimized for JSON APIs.

#rails #ruby #api #backend #json #rest

PNG PDF Copy Prompt

Project Directory

```
myapi/
    Gemfile
    Gemfile.lock
    Rakefile
    config.ru
  > app/
    > controllers/
        application_controller.rb  ActionControlle…
    > api/  Versioned names…
      > v1/  API v1
          base_controller.rb  V1 base
          users_controller.rb
          posts_controller.rb
          auth_controller.rb
    > concerns/
        authenticatable.rb  JWT/token auth
        error_handler.rb
  > models/
      application_record.rb
      user.rb
      post.rb
  > serializers/  JSON serializat…
      user_serializer.rb
      post_serializer.rb
  > services/  Business logic
      auth_service.rb
      user_service.rb
  > config/
      application.rb  API mode config
      routes.rb
      database.yml
      credentials.yml.enc
    > environments/
        development.rb
        test.rb
        production.rb
    > initializers/
        cors.rb  CORS settings
        jwt.rb  Token config
  > db/
      migrate/
      seeds.rb
      schema.rb
  > spec/  RSpec tests
      spec_helper.rb
      rails_helper.rb
    > requests/  API integration…
        api/
      models/
      services/
      factories/  FactoryBot
    lib/
    log/
    public/
    tmp/
    bin/
```

Why This Structure?

Created with `rails new --api`, this strips Rails to essentials for JSON APIs. No views, no asset pipeline, lighter middleware stack. `ActionController::API` replaces `ActionController::Base` for faster responses.

Key Directories

app/controllers/api/v1/ - Versioned API controllers

app/serializers/ - ActiveModelSerializers or fast_jsonapi

app/services/ - Business logic out of controllers

config/initializers/cors.rb - rack-cors configuration

spec/requests/ - API integration tests with RSpec

Versioned API Routes

```ruby
# config/routes.rb
namespace :api do
  namespace :v1 do
    resources :users, only: [:index, :show, :create]
    resources :posts
    post 'auth/login', to: 'auth#login'
  end
end
```

Getting Started

1. `rails new myapi --api`
2. `cd myapi`
3. Add `rack-cors` and `jwt` gems to Gemfile
4. `bundle install`
5. `rails db:create && rails db:migrate`
6. `rails server`

When To Use This

- Backend for React, Vue, or mobile apps
- Microservices that communicate via JSON
- Third-party API consumption
- When you want Rails conventions without frontend

Trade-offs

No views - Need separate frontend deployment

Serialization choice - Must pick a JSON serializer gem

Auth complexity - JWT/token management vs session cookies

Best Practices

- Version your API from day one (`/api/v1/`)
- Use serializers—never `render json: @user`
- Extract auth logic to a concern or service
- Write request specs, not controller specs
- Return proper HTTP status codes