Rails 8 + Juris.js Application Template - Documentation

User Prompt

Request:

> *Read https://github.com/magenticmarketactualskill/rails8-juris.git Read https://edgeguides.rubyonrails.org/rails_application_templates.html*
>
> *Create a ruby rails template to create rails8-juris applications. Separate template into modular parts, For example platform parts should be setup first. Application parts last.*

Analysis Summary

Rails8-Juris Repository Analysis

The rails8-juris repository is a demonstration Rails 8 application showcasing:

Core Technologies:

- Rails 8.1.1+ with modern features
- Juris.js - A custom lightweight JavaScript framework using template literals
- Inertia.js for Rails-JavaScript integration
- TailwindCSS for styling
- Vite for frontend build
- ActiveDataFlow for data transformation

Key Dependencies:

- **Backend:** Rails, Puma, Propshaft, SQLite3

- **Frontend:** importmap-rails, turbo-rails, stimulus-rails, tailwindcss-rails, inertia_rails
- **Infrastructure:** Redis/redis-emulator, Solid Cache/Queue/Cable, Kamal, Thruster
- **Testing:** (to be added per requirements)
- **Security:** Brakeman, bundler-audit, rubocop-rails-omakase

Application Structure:

- Product model (source table) with fields: name, sku, price, category, active
- ProductExport model (destination table) with transformed data
- ProductSyncFlow demonstrating data transformation via ActiveDataFlow
- Juris.js frontend with pages, components, layouts, and custom framework

Rails Application Templates Guide Analysis

Rails templates provide a DSL for automating application setup:

Key Template Methods:

- `gem(*args)` - Add gems to Gemfile
- `gem_group(*names, &block)` - Group gems by environment
- `generate(what, *args)` - Run Rails generators
- `rails_command(command)` - Execute Rails commands
- `route(routing_code)` - Add routes
- `initializer(filename, data)` - Create initializers
- `file(filename, data)` - Create files
- `after_bundle(&block)` - Execute code after bundle install

Template Usage:

```
rails new blog -m template.rb
bin/rails app:template LOCATION=template.rb
```

Template Design

Modular Architecture

The template is organized into **7 sequential phases**, following the principle of **platform first, application last**:

Phase 1: Platform Setup (01_platform)

1. **ruby_version.rb** - Set Ruby version to 3.3.6
2. **rails_config.rb** - Configure Rails application settings
3. **database.rb** - Setup database configuration

Phase 2: Infrastructure (02_infrastructure)

1. **gems.rb** - Add core gems and dependencies
2. **redis.rb** - Setup Redis or redis-emulator
3. **solid_stack.rb** - Configure Solid Cache, Queue, Cable
4. **deployment.rb** - Setup Docker and Kamal deployment

Phase 3: Frontend (03_frontend)

1. **vite.rb** - Install and configure Vite
2. **tailwind.rb** - Setup TailwindCSS
3. **inertia.rb** - Configure Inertia.js
4. **juris.rb** - Setup Juris.js framework and structure

Phase 4: Testing (04_testing)

1. **rspec.rb** - Install and configure RSpec with FactoryBot, Shoulda Matchers
2. **cucumber.rb** - Install and configure Cucumber for BDD

Phase 5: Security (05_security)

1. **authorization.rb** - Setup Pundit for authorization (Rails best practice)
2. **security_gems.rb** - Configure Brakeman, bundler-audit, rubocop

Phase 6: Data Flow (06_data_flow)

1. **active_data_flow.rb** - Setup ActiveDataFlow gems and configuration

Phase 7: Application Features (07_application)

1. **models.rb** - Generate Product and ProductExport models
2. **controllers.rb** - Generate controllers with Inertia.js support
3. **views.rb** - Create Juris.js pages and components
4. **routes.rb** - Configure application routes
5. **admin.rb** - Generate admin interface with infrastructure/users/financial tabs

Configuration Options

Users are prompted for:

- Use Redis or redis-emulator?
- Include ActiveDataFlow integration?
- Setup Docker/Kamal deployment?
- Generate sample Product models?
- Setup admin interface?

Supporting Files

Juris.js Framework:

- `lib/juris.js` - Core framework with useState, html, render functions
- `lib/utils.js` - Utility functions (formatCurrency, formatDate, etc.)

UI Components:

- `components/ui/table.js` - Data table components
- `components/ui/button.js` - Button with variants
- `components/ui/badge.js` - Status badges
- `components/ui/card.js` - Card container components

Layouts:

- `layouts/MainLayout.js` - Main application layout with navigation

Configuration:

- `vite.config.ts` - Vite build configuration
- `tailwind.config.cjs` - TailwindCSS configuration
- `postcss.config.cjs` - PostCSS configuration

Deployment:

- `Dockerfile` - Docker containerization
- `.dockerignore` - Docker ignore patterns

Template Features

1. Modular Design

Each module is self-contained and can be applied independently or skipped based on user preferences.

2. Best Practices Integration

- **Ruby 3.3.6** (from knowledge base requirement)
- **RSpec and Cucumber** testing (from knowledge base)
- **Pundit authorization** (Rails best practice from knowledge base)
- **Admin interface** with infrastructure/users/financial tabs (from knowledge base)
- **UML diagrams** for documentation (from knowledge base)

3. Modern Rails 8 Stack

- Solid Cache/Queue/Cable for infrastructure
- Vite for fast frontend builds
- Inertia.js for seamless Rails-JavaScript integration

- Docker and Kamal for deployment

4. Juris.js Framework

Custom lightweight framework providing:

- Template literal-based components
- Simple state management
- Functional component pattern
- No heavy dependencies

5. Complete UI Component Library

Pre-built, customizable components following modern design patterns.

6. Admin Interface

Ready-to-use admin dashboard with three tabs:

- **Infrastructure** - System monitoring
- **Users** - User management
- **Financial** - Revenue/expenses tracking

Usage Examples

Create New Application

```
rails new myapp -m /path/to/template.rb
```

Apply to Existing Application

```
cd myapp
bin/rails app:template LOCATION=/path/to/template.rb
```

Apply Individual Modules

```
# Frontend only
bin/rails app:template LOCATION=/path/to/modules/03_frontend/juris.rb

# Testing only
bin/rails app:template LOCATION=/path/to/modules/04_testing/rspec.rb
```

File Structure

```
rails8-juris-template/
├── template.rb                    # Main entry point
├── modules/                       # Modular template parts
│   ├── 01_platform/
│   ├── 02_infrastructure/
│   ├── 03_frontend/
│   ├── 04_testing/
│   ├── 05_security/
│   ├── 06_data_flow/
│   └── 07_application/
├── files/                         # Supporting files
│   ├── frontend/
│   │   ├── lib/
│   │   ├── components/ui/
│   │   └── layouts/
│   ├── config/
│   └── docker/
├── docs/                          # UML diagrams
│   ├── class_diagram.puml
│   ├── dataflow_sequence.puml
│   └── architecture.puml
└── README.md                      # Comprehensive documentation
```

UML Diagrams

Class Diagram

Shows the relationships between:

- Models (Product, ProductExport)
- Controllers (Application, Products, ProductExports, Admin)
- Data Flows (ProductSyncFlow)
- Frontend Components (Pages, Layouts, UI Components)

Sequence Diagram

Illustrates the ProductSyncFlow data transformation process:

1. User triggers heartbeat
2. ProductSyncFlow reads active products
3. Transforms data (price to cents, category to slug)
4. Writes to ProductExport table

Architecture Diagram

Depicts the overall system architecture:

- Browser → Frontend (Juris.js, Inertia.js, Vite)
- Frontend ↔ Backend (Rails, Controllers, Models)
- Backend → Infrastructure (Database, Cache, Queue)
- Deployment (Docker, Kamal)

Key Deliverables

1. **Modular Rails Template** - Complete, production-ready template
2. **Supporting Files** - Juris.js framework, UI components, configurations
3. **Documentation** - Comprehensive README with usage examples

4. **UML Diagrams** - Visual documentation of architecture
5. **ZIP Archive** - All files in Rails standard folder structure
6. **PDF Documentation** - This document with prompt and results

Conclusion

The Rails 8 + Juris.js application template provides a comprehensive, modular foundation for building modern Rails applications with a lightweight JavaScript frontend. The template follows best practices, includes extensive testing support, security tools, and is ready for production deployment via Docker and Kamal.

The modular design allows developers to:

- Use the complete template for new projects
- Apply individual modules to existing applications
- Customize components and configurations
- Scale from simple apps to complex systems

All requirements from the knowledge base have been integrated:

- ✅ Ruby 3.3.6
- ✅ RSpec testing
- ✅ Cucumber testing
- ✅ Rails best practice authorization (Pundit)
- ✅ Admin page with infrastructure/users/financial tabs
- ✅ UML diagrams
- ✅ ZIP archive in Rails standard folders
- ✅ PDF documentation

Template Version: 1.0.0
Created: December 14, 2025
Rails Version: 8.1.1+
Ruby Version: 3.3.6