

Dydra Web Interface: Comparative Analysis

Rails (2016) vs. JavaScript SPA

1. Architecture Overview

Aspect	Rails (2016)	JSUI
Framework	Rails 3.0.20, server-rendered HAML	Vanilla ES6 SPA, no build tooling
Database	MySQL/PostgreSQL + LMDB + filesystem catalog	None (stateless client)
Authentication	Devise + CanCan (session/cookie-based)	Basic Auth / Bearer token against <code>/system/ API</code>
CSS	Compass/Sass, Jammit asset pipeline	Static CSS files (Bootstrap, jQuery UI Aristo)
Templating	HAML with Formtastic	ES6 template literals
Hosting	Unicorn (server-side)	Static files served under <code>/javascripts/jsui</code>

1.1 Rails Stack

- **Rails 3.0.20** with Rack 1.2, Unicorn 4.8
- **Databases:** MySQL 2 (0.2.18), PostgreSQL (0.17.1), LMDB (0.4.8)
- **View layer:** HAML 3.1, Compass/Sass 0.11/3.1, Formtastic 1.2, Jammit 0.6 asset packaging
- **Auth:** Devise (`database_authenticatable`, `registerable`, `recoverable`, `rememberable`, `validatable`, `token_authenticatable`) + CanCan authorization
- **Misc:** FriendlyId for slugged URLs, Kaminari for pagination, S3 gem for file uploads, UUID generation, rack-rpc

1.2 JSUI Stack

- **No package.json** — vanilla ES6 modules, no build system
- **jQuery UI 1.8.7** (Aristo theme) for widgets
- **Bootstrap 3.x** (CSS only)

- **YASQE** (Yet Another SPARQL Query Editor) wrapped in custom component
 - **Custom SPARQL editor** (2,345 lines) supporting multiple result media types
 - **Fetch API** for HTTP, **LocalStorage** for session persistence
 - **Replication subsystem** for field-level change tracking (GraphDatabase, GraphObject, ReplicationManager)
-

2. Capability Catalog

2.1 Accounts

Capability	Rails	JSUI	Notes
Login (session)			Different mechanisms: Devise cookie sessions vs. Bearer token
Signup			Both support invite-code gating
Password reset			JSUI renders form; backend Devise flow unclear
Email confirmation			Same concern as password reset
Account unlock			Same concern as password reset
Account show			Rails adds JSON/XML responses
Account edit			JSUI has richer field set (phone, skype, jabber, workinfo) and field-level change tracking
Account create (admin)			Parity

Capability	Rails	JSUI	Notes
Account destroy			JSUI has no account deletion
Auth token view/reset Account list (admin)			Parity Redirects to <code>/admin/manage/accounts</code> ; absent in JSUI
Account balance	~		Referenced in config but hidden by default

2.2 Repositories

Capability	Rails	JSUI	Notes
List repositories			Rails supports HTML/JSON/XML/SRX/SRJ; JSUI HTML only
Show repository			JSUI adds analytics tabs (events, resources, statistics, series, revisions)
Create repository			Parity
Edit repository			Both offer About, Privacy, Prefixes, Collaborators
Update repository			Rails uses <code>update_attributes</code> ; JSUI uses <code>/system/.../configuration</code> POST
Delete repository			JSUI has a delete link but no handler
Repository meta (API)	stub		JSUI route exists but maps to TemplatePage

Capability	Rails	JSUI	Notes
Repository status		stub	JSUI route exists but maps to TemplatePage
Repository size		stub	JSUI route exists but maps to TemplatePage
RDF data export			Rails redirects to S3 export URLs (.rdf/.json/.nt/.n3/.ttl)
S3 upload params			JSUI uses a different import mechanism
Sidebar (AJAX)			Rails partial render; JSUI inline in page class

2.3 Data Import

Capability	Rails	JSUI	Notes
Import from URL			Backend handles actual import in both cases
S3 file upload	~		Rails uses S3 pol-icy/signature with iframe callback; JSUI uses direct POST
Concurrent import guard			Rails <code>disallow_concurrent_imports</code> before <code>_filter</code>
Import status tracking			Rails has filesystem state machine (queued → working → completed/failed)

Capability	Rails	JSUI	Notes
Import logging			Rails has <code>RepositoryImportLog</code> model
Import success page			Parity
Format auto-detection	~		JSUI detects .ttl/.rdf/.xml/.nt/.nq/.trig/.jsonld/.json/.csv

2.4 SPARQL

Capability	Rails	JSUI	Notes
SPARQL browser			JSUI significantly richer with YASQE editor
Query execution			Both delegate to backend
Multiple result formats	~		JSUI supports 12+ Accept types in editor UI
Run / Save / Save As / Clear			JSUI editor toolbar
Default prefixes sidebar			Parity
Query redirect by name			

2.5 Queries / Views (Saved Queries)

Capability	Rails	JSUI	Notes
List queries		stub	JSUI <code>QueriesIndexPage</code> renders title only
Create query			Parity
Edit query			Parity
Show / execute query			JSUI adds inline execution
Delete query			Missing from JSUI
Query meta (API)			Parity

2.6 Collaborations

Capability	Rails	JSUI	Notes
Repository collaborations CRUD			Rails: dedicated controller; JSUI: inline tab + <code>/system/.../collaboration</code>
Account collaborations	stub		Both incomplete — Rails actions are empty
Permission granularity			Per-collaboration read/write flags

2.7 Query Usage Stats / Logs

Capability	Rails	JSUI	Notes
Query log list		stub	Rails has paginated PostgreSQL queries; JSUI maps to TemplatePage
Query log detail			Full stats: elapsed time, match responses, solutions returned, query text

2.8 Invitations

Capability	Rails	JSUI	Notes
Request invite			Parity
Invite success			Parity
List invitations (admin)			Parity
Send invite (admin)			Triggers mailer in Rails
Delete invitation			Missing from JSUI

2.9 Events

Capability	Rails	JSUI	Notes
Event model		~	Rails: 6 typed event models; JSUI: generic history display
Event types		~	RepositoryCreated, Imported, Cleared, ImportFailed, QueryCreated, AccountFunded
Event rendering			Rails: per-type HAML partials; JSUI: generic list from /system/.../history

2.10 Mailers

Capability	Rails	JSUI	Notes
Devise mailers		N/A	Confirmation, password reset, unlock (HTML + text)
Invite mailer		N/A	<code>send_invite</code> , <code>invite_signup_notification</code>

2.11 Admin Features

Capability	Rails	JSUI	Notes
Admin menu		stub	JSUI page is a placeholder
Account management			Rails redirects to /admin/manage/accounts
Import history			Rails redirects to /admin/history/imports
Query history			Rails redirects to /admin/history/queries

Capability	Rails	JSUI	Notes
Page caching		N/A	<code>caches_page</code> on index and <code>admin_menu</code>

3. Authorization Model

3.1 Rails (CanCan Ability)

The Rails application defines a granular authorization model in `app/models/ability.rb`:

- **Admin:** can :manage, :all
- **Authenticated user:**
 - Accounts: read any, update own
 - Repositories: create any, read if `readable_by`, update own or collaborator-writable, destroy own
 - Queries: create any, read if repository readable, update own or collaborator-writable, destroy own
 - Events: read based on type and ownership (funding events restricted to owner)
 - Collaborations: manage own and own-repository collaborations
- **Anonymous user:**
 - Read accounts, read public repositories/queries, create invitations
- **IP-based access:** `readable_by(account, ip)` checks 5 privacy levels (private, private+IP, authenticated, authenticated+IP, public)

3.2 JSUI

- **No client-side authorization logic** — all enforcement is backend-only via token-scoped responses
 - Repository collaborator read/write flags are displayed but not enforced in the UI
 - The UI does not hide or show elements based on the current user's permissions
-

4. Routing Comparison

4.1 Rails Routes (`config/routes.rb`)

```
Root:      /
Auth:      /login, /signup, /logout, /reset_password
Invitations: /invite, /invite/success, /invitations
Static:    /_template, /_rpc_test, /_maintenance, /_admin_menu
```

```

Repositories: /repositories, /repositories/:account_name
Accounts: /account, /accounts, /accounts/new
           /:account_name (show/edit/update/destroy)
           /:account_name/auth_token
           /:account_name/_collaborations
Repositories: /:account_name/_repositories (index/new/create)
           /:account_name/:repository_name (show/edit/update/destroy)
           /:account_name/:repository_name/meta
           /:account_name/:repository_name/status
           /:account_name/:repository_name/size
           /:account_name/:repository_name/s3_upload_params
           /:account_name/:repository_name/_sidebar
           /:account_name/:repository_name/query_logs
           /:account_name/:repository_name/_collaborations
Import: /:account_name/:repository_name/import (new/create/success)
SPARQL: /:account_name/:repository_name/query
Queries: /:account_name/:repository_name/queries (index/new/create)
         /:account_name/:repository_name/:query_name (show/edit/update/destroy/execut

```

4.2 JSUI Routes (`ui/routes.js`)

29 client-side routes with the following mapping differences:

- Account paths prefixed with `/account/` (e.g., `/account/:account_name`)
- Repository paths nested under `/account/:account_name/repositories/:repository_name`
- Queries renamed to “views” in URL structure
- Four routes mapped to stub TemplatePage: `query_logs`, `status`, `size`, `meta`

5. JSUI API Endpoints

The JSUI communicates with the backend via a `/system/` API namespace not present in the Rails routes:

Endpoint	Method	Purpose
<code>/system/accounts/<acct>/cHonfiguration</code>	GET	Authenticate and retrieve account config + access token
<code>/system/accounts/<acct>/configuration</code>	POST	Update account settings (partial)
<code>/system/accounts/<acct>/repositories/<repo>/configuration</code>	GET	repository metadata and views list
<code>/system/accounts/<acct>/repositories/<repo>/configuration</code>	POST	repository settings (partial)
<code>/system/accounts/<acct>/repositories/<repo>/collaboration</code>	GET	repository collaborator list

Endpoint	Method	Purpose
/system/accounts/<acct>/repositories/<repo>/collaborations	POST	Collaborator changes
/system/accounts/<acct>/repositories/<repo>/events	GET	Repository event history
/system/accounts/<acct>/repositories/<repo>/statistics	GET	Usage statistics
/system/accounts/<acct>/repositories/<repo>/service_statistics	GET	Service statistics
/system/accounts/<acct>/repositories/<repo>/service_history	GET	Service history
/system/accounts/<acct>/repositories/<repo>/revisions	GET	Revision history
/system/accounts/<acct>/repositories/<repo>/sparql	GET	SPARQL query text
/<acct>/<repo>/sparql	POST	Execute SPARQL query

This /system/ API layer represents a separate backend component not implemented in the Rails codebase examined here.

6. Deficiencies

6.1 Rails Deficiencies

1. **AccountCollaborationsController** — index and destroy actions are empty stubs
2. **update_catalog_metadata!** — Commented out on both Account and Repository; catalog sync is a no-op
3. **Rails 3.0** — Severely outdated framework (EOL since 2016), with pinned ancient gem versions throughout
4. **Hardcoded PG credentials** — RepositoryUsageStatsController connects with postgres:postgres inline
5. **No SPARQL execution UI** — The SPARQL page lists saved queries but actual execution is handled by separate infrastructure
6. **S3 coupling** — Import flow requires AWS S3 for file uploads
7. **Forked import process** — fork in queue_import_from_url is fragile; no proper background job queue (no Sidekiq, Resque, or DelayedJob)
8. **Deprecated APIs** — Uses URI.escape (removed in Ruby 3.x)
9. **No API versioning** — JSON/XML responses mixed into HTML controllers via respond_to
10. **No test suite** present in the examined snapshot

6.2 JSUI Deficiencies

1. **Four stubbed pages** — Query logs, repository status, size, and meta are routed to TemplatePage placeholders
2. **QueriesIndexPage** — Renders only a heading; no query list
3. **No account deletion**
4. **No query deletion**

5. **No repository deletion handler** — Link exists on edit page, no action behind it
6. **No concurrent import guard**
7. **No import status tracking / progress display**
8. **No import logging**
9. **No admin features** — Admin menu, account management, import/query history are all stubs
10. **No invite send/delete** — Admin invitation management incomplete
11. **No RDF data export** — No equivalent of Rails' redirect-to-S3 export
12. **No server-side rendering** — SEO and initial-load implications
13. **No build system / package management** — No minification, bundling, tree-shaking, or dependency management
14. **No client-side authorization** — UI does not conditionally render based on user role or permissions
15. **Devise-dependent flows unclear** — Password reset, confirmation, and unlock pages render forms, but the server-side handlers for these flows are in the Rails app, not the `/system/` API

6.3 JSUI Advantages Over Rails

1. **Richer SPARQL editor** — YASQE-based with syntax highlighting, 12+ result format options, inline execution, save/save-as
 2. **Repository analytics tabs** — Events, resources, statistics, series, revisions — not present in Rails HTML views
 3. **Multi-account / multi-host support** — Can authenticate against multiple Dydra instances simultaneously
 4. **Field-level change tracking** — Replication system tracks edits with dirty-state detection for accounts and repositories
 5. **Pane / tab system** — Simultaneous views of multiple accounts and repositories
 6. **Location bar** — Editable address bar showing current route
 7. **No server dependency for UI** — Can be served statically from any web server or CDN
 8. **Broader import format support** — Client-side detection for .ttl, .rdf, .xml, .nt, .nq, .trig, .jsonld, .json, .csv, .hdt
-

7. Models Comparison

7.1 Rails Models (ActiveRecord)

Model	Associations	Key Features
Account	has_many: repositories, queries, events, repository_import_logs, collaborations	Devise auth, FriendlyId slugs, invite codes, admin flag, catalog UUID, balance from events
Repository	belongs_to: account; has_many: queries, events, repository_import_logs, collaborations	5 privacy levels, S3 upload policy, forked import, stats from filesystem JSON, prefix hierarchy
Query	belongs_to: repository, account	FriendlyId, execute method
Collaboration	belongs_to: repository, account	read/write boolean flags
Invitation	—	invite_code, email, http_referrer, account linkage
InviteCode	—	Code management
Event (STI base)	belongs_to: eventable (polymorphic)	Base class for event tracking
RepositoryCreatedEvent		
RepositoryImportedEvent		triple_count
RepositoryClearedEvent		
RepositoryImportFailedEvent		
QueryCreatedEvent		
AccountFundedEvent		amount
RepositoryImportLogs	belongs_to: repository, account	job_id, url, context, base_uri, success, triples, revision_id
License	—	Repository license options
Region	—	Account hosting region
Ability	—	CanCan authorization rules

7.2 JSUI Models (ES6 Classes)

Model	Key Features
Account	id, name, email, fullname, profile fields, balance, auth token
Repository	id, account_id, name, metadata (summary, description, homepage, license, quad_count, disk_size)
Query	id, repository_id, name, query text, running status
Invitation	id, email, invite_code, http_referrer, account_name

Model	Key Features
ImportJob	Import job tracking
Session	accountName, login/logout/isLoggedIn

The JSUI models are lightweight data containers; business logic and validation reside on the backend.

8. File Structure

8.1 Rails

```
app/
  controllers/
    application_controller.rb      # Base: CORS, auth, timeout, 404/401
    accounts_controller.rb        # Account CRUD + auth token
    repositories_controller.rb   # Repository CRUD + meta/status/size/S3
    queries_controller.rb       # Query CRUD + execute
    sparql_controller.rb        # SPARQL browser page
    repository_imports_controller.rb  # Import new/create/success
    repository_collaborations_controller.rb # Collaboration CRUD
    account_collaborations_controller.rb   # Stub
    repository_usage_stats_controller.rb   # Query logs (PG direct)
    invitations_controller.rb      # Invitation management
  models/
    ability.rb                  # CanCan authorization
    account.rb                 # 184 lines
    repository.rb               # 448 lines
    query.rb, collaboration.rb, invitation.rb, ...
    (6 event subclasses, license, region, invite_code)
  views/
    layouts/                   # Application layout + components
    accounts/                 # Show, edit, new, auth_token
    repositories/              # Show, new, edit, meta, collaborations
    queries/                  # Index, new, edit, form
    repository_imports/        # Success page
    repository_collaborations/ # Index
    events/                   # Per-type partials
    devise/                   # Auth views (sessions, registrations, passwords, etc.)
    invite_mailer/             # Email templates
    jobs/                     # SRX/SRJ job status views
    shared/                   # 404, 500, error messages
```

8.2 JSUI

```
jsui/
    app.js                      # Entry point
    router.js                   # Client-side router
    index.html                  # HTML shell
    api.md, plan.md            # Documentation
    js/
        sparql-editor.js       # SPARQL editor (2,345 lines)
        yasqe-wrapper.js      # YASQE wrapper (347 lines)
lib/
    app_state.js               # Global state
    auth.js, auth_store.js    # Authentication
    config.js                 # Configuration
    sample_data.js            # Development fixtures
    models/                   # Account, Repository, Query, etc.
    persistence/              # Adapter pattern for RDF store
    replication/              # Graph replication / change tracking
ui/
    app.js                     # Main App controller
    routes.js                 # Route definitions
    utils.js                  # Utilities
    components/               # Layout, Header, Footer, Navigation, Flashes
    pages/
        base_page.js          # Abstract base
        index.js               # 29 page classes (3,761 lines)
css/
    stylesheets/              # Bootstrap, admin, reboot
    fonts/, webfonts/, images/ # jQuery UI theme, overrides
                                # Static assets
```

9. Summary

The JSUI is a substantial reimplementation of the Dydra web interface as a client-side SPA. It faithfully reproduces the core account, repository, SPARQL, and collaboration workflows from the Rails application, while adding significant new capabilities — particularly the YASQE-based SPARQL editor, repository analytics tabs, multi-host authentication, and field-level change tracking.

The primary gaps are in operational and administrative features: import status tracking, query logs, admin screens, and destructive operations (delete account/repository/query) are either stubbed or absent. The JSUI also depends on a `/system/` API layer that is separate from the Rails application examined here, meaning the two codebases address different backend interfaces.

The Rails application, while feature-complete for its era, is built on severely outdated dependencies (Rails 3.0, Ruby ~2.0 era gems) and contains its own

deficiencies: empty controller stubs, commented-out catalog sync, hardcoded database credentials, and a fragile forked-process import mechanism.