Dydra Web Interface: Comparative Analysis (Revised)

Rails (2016) vs. JavaScript SPA — February 4, 2026

This revision updates the February 1 analysis to reflect the addition of a complete admin SPA, repository/account/view destructive operations, RDF data export, and cross-app navigation.

---

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 `/system/` API
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 `/javascripts/jsui`
Admin	Integrated into same Rails app (CanCan admin role)	Separate SPA at `admin.html` with dedicated entry point

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 authoriza-

tion
- **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)
- **Two SPA entry points**: `index.html` (user app) and `admin.html` (admin app), sharing router, auth, state, and layout infrastructure

---

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

Capability	Rails	JSUI	Notes
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 — JSUI admin SPA has New Account dialog
Account destroy (admin)			**New** — JSUI admin SPA has delete with confirmation + N-Triples result dialog
Auth token view/reset			Parity
Account list (admin)			**New** — JSUI admin SPA fetches from `/system/accounts`, sortable with check-all
Account details (admin)			**New** — JSUI admin SPA fetches configuration and displays in modal
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			User app + admin SPA (New Repository dialog with account selector)
Edit repository			Both offer About, Privacy, Prefixes, Collaborators
Update repository			Rails uses `update_attributes`; JSUI uses `/system/.../configuration` POST
Delete repository			**New** — User app `handleRepositoryDelete` with confirmation; admin SPA also has delete
Clear repository			**New** — `handleRepositoryClear` removes all triples with confirmation
Export repository	~		**New** — `handleRepositoryExport` supports multiple RDF serializations

Capability	Rails	JSUI	Notes
Repository meta (API)		stub	JSUI route exists but maps to TemplatePage
Repository status		stub	JSUI route exists but maps to TemplatePage
Repository size		stub	JSUI route exists but maps to TemplatePage
S3 upload params			JSUI uses a different import mechanism (direct POST)
Sidebar (AJAX)			Rails partial render; JSUI inline in page class
List all repos (admin)			**New** — admin SPA lists repos filtered by selected accounts

2.3 Data Import

Capability	Rails	JSUI	Notes
Import from URL			Backend handles actual import in both cases
S3 file upload		~	Rails uses S3 policy/signature with iframe callback; JSUI uses direct POST
Concurrent import guard			Rails `disallow_concurrent_imports` before_filter

Capability	Rails	JSUI	Notes
Import status tracking			Rails has filesystem state machine (queued → working → completed/failed)
Import logging			Rails has `RepositoryImportLog` model
Import success page			Parity
Format auto-detection	~		JSUI detects .ttl/.rdf/.xml/.nt/.nq/.trig/.jsonld/.json/.csv/.

2.4 Data Export

Capability	Rails	JSUI	Notes
RDF export			**New** — Rails redirects to S3; JSUI fetches directly with content negotiation
Export format selection	~		JSUI offers Turtle, N-Triples, N-Quads, RDF/XML, RDF/JSON, JSON-LD, TriG
Export as file download			JSUI creates Blob URL + click-to-download

2.5 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			

2.6 Queries / Views (Saved Queries)

Capability	Rails	JSUI	Notes
List queries		stub	JSUI `QueriesIndexPage` renders title only
Create query			Parity
Edit query			Parity
Show / execute query			JSUI adds inline execution
Delete query/view			**New** — `handleViewDelete` with confirmation dialog
Query meta (API)			Parity

2.7 Collaborations

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

2.8 Query Usage Stats / Logs

Capability	Rails	JSUI	Notes
Query log list (per-repo)		stub	Rails has paginated PostgreSQL queries; JSUI user app maps to TemplatePage
Query log detail			Full stats: elapsed time, match responses, solutions returned, query text
Query history (admin, multi-repo)	~		**New** — admin SPA aggregates `service_history` across selected repos, sortable columns
Transaction history (admin)			**New** — admin SPA fetches from `/system/service_history/transactions`, sortable

2.9 Invitations

Capability	Rails	JSUI	Notes
Request invite			Parity
Invite success			Parity
List invitations (admin)			**New** — admin SPA lists with email, referrer, code, account columns
Send invite (admin)			**New** — admin SPA POSTs to `/invitations`
Delete invitation (admin)			**New** — admin SPA DELETEs with confirmation
Create invitation (admin)			**New** — admin SPA has New Invitation dialog

2.10 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`

2.11 Mailers

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

2.12 Admin Features

Capability	Rails	JSUI	Notes
Admin login with privilege check			**New** — JSUI verifies `administrator_of` via `/system/users/{acct}/configuration`
Admin dashboard			**New** — tab-based dashboard with lazy-loaded panes
Account management (list/create/delete)			**New** — full CRUD with check-all, refresh, details modal
Repository management (list/create/delete)			**New** — filtered by selected accounts, with create/delete dialogs
Invitation management			**New** — list, send, delete, create
Query history			**New** — multi-repo aggregation with sortable columns
Transaction history			**New** — JSUI-only feature; not in Rails

Capability	Rails	JSUI	Notes
Import history			Rails redirects to `/admin/history/imports`; not yet implemented in JSUI
Payment history			Rails has payment tracking; JSUI PaymentHistoryPage redirects to dashboard
Page caching		N/A	`caches_page` on index and admin_menu
Cross-app navigation	N/A		**New** — Admin link in user header opens admin SPA; User link in admin header returns to user SPA

---

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
- **Admin verification** is performed at login only — the admin SPA checks `administrator_of` but does not re-verify on subsequent requests

---

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/execu
```

4.2 JSUI User 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

4.3 JSUI Admin Routes (`ui/admin/routes.js`)

10 admin-specific routes, all under the `/admin` prefix:

```
/login                              → AdminLoginPage
/                                   → AdminDashboardPage (auth-gated)
/manage/accounts                    → ManageAccountsPage
/manage/accounts/:account_name      → ManageAccountPage
/manage/repositories                → ManageRepositoriesPage
/invitations                        → AdminInvitationsPage
/invite                             → AdminInviteNewPage
/history/queries                    → QueryHistoryPage
/history/transactions               → TransactionHistoryPage
/logout                             → logout handler
```

All admin routes except `/login` require an authenticated session; the dashboard route redirects to login if unauthenticated. Each subpage class extends `AdminDashboardPage` and auto-selects the appropriate tab.

---

5. JSUI API Endpoints

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

5.1 User App Endpoints

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

Endpoint	Method	Purpose
`/system/accounts/<acct>/repositories/<repo>/collaboration`	POST	Save collaborator changes
`/system/accounts/<acct>/repositories/<repo>/history`	GET	Repository event history
`/system/accounts/<acct>/repositories/<repo>/storage`	GET	Disk usage statistics
`/system/accounts/<acct>/repositories/<repo>/service_statistics`	GET	Service statistics
`/system/accounts/<acct>/repositories/<repo>/service_history`	GET	Statistics over time
`/system/accounts/<acct>/repositories/<repo>/revisions`	GET	RDF revision history
`/system/accounts/<acct>/repositories/<repo>/views/<view>`	GET	Retrieve SPARQL query text
`/system/accounts/<acct>/repositories/<repo>/views/<view>`	DELETE	**New** — Delete a saved view
`/system/accounts/<acct>/repositories/<repo>`	DELETE	**New** — Delete a repository
`/<acct>/<repo>/sparql`	POST	Execute SPARQL query
`/<acct>/<repo>`	GET	**New** — Export repository (content-negotiated RDF format)

5.2 Admin App Endpoints

Endpoint	Method	Purpose
`/system/users/<acct>/configuration`	GET	Verify admin privileges (`administrator_of`)
`/system/accounts`	GET	List all system accounts
`/system/accounts`	POST	Create new account
`/system/accounts/<acct>`	DELETE	Delete account (returns N-Triples)
`/system/accounts/<acct>/configuration`	GET	Fetch account details
`/system/accounts/<acct>/repositories`	GET	List account's repositories
`/system/accounts/<acct>/repositories`	POST	Create new repository
`/system/accounts/<acct>/repositories/<repo>`	DELETE	Delete repository (returns N-Triples)
`/system/accounts/<acct>/repositories/<repo>/service_history`	GET	Query performance history
`/system/service_history/transactions`	GET	System-wide transaction history
`/invitations`	GET	List invitations
`/invitations`	POST	Send/create invitation
`/invitations/<email>`	DELETE	Delete invitation

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
11. **No transaction history** — No equivalent of the JSUI admin's transaction history view

6.2 JSUI Deficiencies

1. **Four stubbed user-app 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~~ **Resolved** — Admin SPA has account deletion with confirmation
4. ~~No query deletion~~ **Resolved** — `handleViewDelete` with confirmation dialog
5. ~~No repository deletion handler~~ **Resolved** — `handleRepositoryDelete` with confirmation dialog
6. **No concurrent import guard**
7. **No import status tracking / progress display**
8. **No import logging**
9. ~~No admin features~~ **Resolved** — Full admin SPA with 5 tabbed sections
10. ~~No invite send/delete~~ **Resolved** — Admin SPA has send, delete, and create

11. ~~No RDF data export~~ **Resolved** — `handleRepositoryExport` with format selection
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
16. **No import history (admin)** — Rails has import history; JSUI admin SPA does not
17. **No payment history (admin)** — `PaymentHistoryPage` exists but redirects to dashboard

6.3 Summary of Resolved Deficiencies (since Feb 1)

Former Deficiency	Resolution
No admin features (stub only)	Complete admin SPA with login, privilege verification, 5-tab dashboard
No account deletion	Admin SPA: DELETE with N-Triples result dialog
No account listing (admin)	Admin SPA: sortable list with check-all, refresh
No repository deletion handler	User app: `handleRepositoryDelete`; Admin SPA: delete button
No query/view deletion	User app: `handleViewDelete` with confirmation
No invite send/delete (admin)	Admin SPA: send, delete, and create invitation
No RDF data export	User app: `handleRepositoryExport` with format picker

6.4 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 mul-

tiple 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
9. **RDF export with format selection** — Multiple serialization formats via content negotiation
10. **Repository clear** — One-click removal of all triples with confirmation
11. **Transaction history** — System-wide transaction monitoring not available in Rails
12. **Admin SPA with filtered views** — Account checkbox filtering cascades to repositories and query history
13. **Sortable admin tables** — Column-header click sorting with directional indicators
14. **Lazy-loaded admin tabs** — Content fetched on demand, reducing initial load
15. **Cross-app navigation** — Bidirectional links between user and admin SPAs
16. **SVG branding** — Vector logos (`Dydra Logo-user.svg`, `Dydra Logo-admin.svg`) with PNG fallbacks

---

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

Model	Associations	Key Features
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
RepositoryImportLog	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
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/
  index.html                        # User SPA HTML shell
  admin.html                        # Admin SPA HTML shell (new)
  app.js                            # User app entry point
  admin-app.js                      # Admin app entry point (new)
  router.js                         # Client-side router (shared)
  api.md, plan.md                   # Documentation
  doc/
     analysis-20260201.md           # Previous comparative analysis
     analysis-20260204.md           # This document
```

```
    implementation.md              # Implementation notes
    requirements.md                # Requirements specification
    session-storage.md             # Session storage documentation
js/
    sparql-editor.js               # SPARQL editor (2,345 lines)
    yasqe-wrapper.js               # YASQE wrapper (347 lines)
lib/
    app_state.js                   # Global state (shared)
    auth.js                        # Authentication (shared)
    auth_store.js                  # Token storage (shared)
    config.js                      # Configuration (shared)
    sample_data.js                 # Development fixtures
    models/                        # Account, Repository, Query, Invitation, ImportJob, S
    persistence/                   # Adapter pattern for RDF store
    replication/                   # Graph replication / change tracking
ui/
    app.js                         # User App controller
    routes.js                      # User route definitions (29 routes)
    utils.js                       # Shared utilities (escapeHtml, joinHtml)
    components/                    # Layout, Header, Footer, Navigation, Flashes
    pages/
        base_page.js               # Abstract base page
        index.js                   # 24 user page classes (~2,800 lines)
    admin/                         # Admin SPA (new)
        app.js                     # Admin App controller
        layout.js                  # Admin layout view
        routes.js                  # Admin route definitions (10 routes)
        pages.js                   # 9 admin page classes (~1,370 lines)
css/                               # Bootstrap, admin, reboot
stylesheets/                       # jQuery UI theme, overrides
images/                            # Logos (SVG + PNG), icons, backgrounds
    Dydra Logo-admin.svg           # Admin logo (new)
    Dydra Logo-user.svg            # User logo (new)
    logo-admin.png                 # Admin logo PNG fallback (new)
    logo.png                       # User logo PNG fallback
    trash.svg                      # Delete icon (new)
    (other static assets)
fonts/, webfonts/                  # Font assets
signup.html, reset_password.html   # Static auth pages
favicon.ico, apple-touch-icon.png  # Browser icons
```

9. Change Log (Feb 1 → Feb 4)

New Files

- `admin.html` — Admin SPA entry HTML with SVG logo, styled login page
- `admin-app.js` — Admin app bootstrap module
- `ui/admin/app.js` — AdminApp class with layout, routing, logout
- `ui/admin/layout.js` — AdminLayoutView with admin-specific navigation
- `ui/admin/routes.js` — 10 admin routes
- `ui/admin/pages.js` — 9 admin page classes (1,370 lines): login with privilege verification, dashboard with lazy-loaded tabs, account/repository/invitation CRUD, query/transaction history with sortable columns
- `ui/utils.js` — Shared utilities extracted for cross-SPA reuse
- `images/Dydra Logo-admin.svg` — Admin logo (bold "DYDRA ADMIN")
- `images/Dydra Logo-user.svg` — User logo (bold "DYDRA")
- `images/logo-admin.png` — PNG fallback for admin logo
- `images/trash.svg` — Delete icon for admin tables

Modified Files

- `ui/pages/index.js` — Added `handleRepositoryDelete`, `handleRepositoryExport`, `handleViewDelete`, `handleRepositoryClear` with full implementations; exported shared helpers
- `ui/app.js` — Added Admin nav link (opens `admin.html` in new window)
- `stylesheets/jsui-overrides.css` — SVG logo background with PNG fallback
- `images/logo.png` — Regenerated from `Dydra Logo-user.svg`

---

10. Summary

The JSUI has progressed from a partial reimplementation of the Dydra web interface to a substantially complete replacement. The addition of the admin SPA closes the largest gap identified in the February 1 analysis: administrative features are now fully operational with account management, repository management, invitation handling, query performance history, and transaction monitoring.

Resolved since Feb 1: All destructive operations (delete account, repository, view), RDF data export, admin dashboard with 5 functional tabs, invitation send/delete, cross-app navigation, and branded SVG logos for both user and admin interfaces.

Remaining gaps: Import workflow polish (concurrent guard, status tracking, logging), four stubbed per-repository information pages (query_logs, status, size, meta), QueriesIndexPage listing, import history (admin), payment history (admin), client-side authorization, and Devise-dependent authentication flows.

The Rails application remains relevant only as a reference for the Devise authentication flows (mailers, confirmation, unlock) and the import status tracking state machine. All other functionality is now replicated or exceeded by the JSUI.