Dydra JSUI Summary

February 5, 2026

---

1. Overview

The JSUI is a complete single-page application replacement for the Dydra web interface, comprising two SPAs:

- **User App** (`index.html`): Account management, repository operations, SPARQL queries, data import/export
- **Admin App** (`admin.html`): System administration with account/repository management, invitation handling, and history monitoring

Both applications share core infrastructure (router, authentication, state management) and communicate with the SPOCQ backend via a RESTful `/system/` API.

---

2. Key Capabilities

2.1 Repository Management

- Create, edit, delete repositories
- Clear all triples with confirmation
- Export to multiple RDF formats (Turtle, N-Triples, N-Quads, RDF/XML, JSON-LD, TriG, CSV)
- Import with progress tracking, byte counts, and abort capability
- Asynchronous import option (`AcceptAsynchronous: notify` header)
- Collaborator management with read/write permissions

2.2 SPARQL Interface

- YASQE-based editor with syntax highlighting
- 12+ result format options via content negotiation
- Save/Save As/Clear query operations
- Saved views with execute and delete actions
- Query history with signature links to query text

2.3 Administration

- Account listing with bulk selection, details modal, create/delete
- Repository management filtered by selected accounts
- Invitation management (list, send, delete, create)
- Query history aggregated across selected repositories
- Transaction history (system-wide)

- Import history (system-wide)
- Role-based UI: non-admin users see data but not destructive controls

2.4 User Experience

- Multi-account/multi-host authentication support
- Pane/tab system for simultaneous views
- Editable location bar showing current route
- Cross-app navigation between user and admin SPAs
- Field-level change tracking with dirty-state detection
- Lazy-loaded admin tabs for reduced initial load

---

3. Advantages Over Rails Implementation

1. **Richer SPARQL editor** — YASQE-based with syntax highlighting, inline execution, multiple result formats
2. **Repository analytics** — Events, resources, statistics, series, revisions tabs
3. **Multi-account support** — Simultaneous authentication against multiple Dydra instances
4. **Field-level change tracking** — Replication system tracks edits with dirty-state detection
5. **Pane/tab system** — View multiple accounts and repositories simultaneously
6. **Location bar** — Editable address bar showing and accepting route input
7. **Static deployment** — No server dependency for UI; can be served from any web server or CDN
8. **Broader import formats** — 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
15. **Cross-app navigation** — Bidirectional links between user and admin SPAs
16. **SVG branding** — Vector logos with PNG fallbacks

---

4. Data Models

4.1 Account

Field	Description
id	Unique identifier
name	Account name (slug)
email	Contact email
fullname	Display name
phone, skype, jabber	Contact fields
workinfo	Work information
balance	Account balance
authentication_token	API access token

4.2 Repository

Field	Description
id	Unique identifier
account_id	Owning account
name	Repository name (slug)
summary	Short description
description	Full description
homepage	External URL
license	License identifier
quad_count	Number of quads stored
disk_size	Storage used

4.3 Query/View

Field	Description
id	Unique identifier
repository_id	Parent repository
name	Query name (slug)
query	SPARQL query text
running	Execution status flag

4.4 Invitation

Field	Description
id	Unique identifier
email	Invitee email

Field	Description
invite_code	Generated code
http_referrer	Referral source
account_name	Linked account (if accepted)

4.5 Session

Field	Description
accountName	Logged-in account
login()	Start session
logout()	End session
isLoggedIn()	Check status

---

5. API Endpoints

5.1 Authentication

Endpoint	Method	Purpose
`/system/accounts/{acct}/configuration`	POST	Authenticate, retrieve token
`/system/users/{acct}/configuration`	GET	Check admin privileges

5.2 Accounts

Endpoint	Method	Purpose
`/system/accounts`	GET	List all accounts
`/system/accounts`	POST	Create account
`/system/accounts/{acct}`	DELETE	Delete account
`/system/accounts/{acct}/configuration`	GET/POST	Read/update settings
`/system/accounts/{acct}/password-reset`	POST	Password reset
`/system/accounts/{acct}/token-reset`	POST	Reset auth token

5.3 Repositories

Endpoint	Method	Purpose
`/system/accounts/{acct}/repositories`	GET	List repositories
`/system/accounts/{acct}/repositories`	POST	Create repository

Endpoint	Method	Purpose
`/system/accounts/{acct}/repositories/{repo}`	DELETE	Delete repository
`/system/accounts/{acct}/repositories/{repo}/configuration`	GET/POST	Settings
`/system/accounts/{acct}/repositories/{repo}/collaborators`	GET/POST	Collaborators
`/system/accounts/{acct}/repositories/{repo}/history`	GET	History
`/system/accounts/{acct}/repositories/{repo}/storage`	GET	Disk usage
`/system/accounts/{acct}/repositories/{repo}/query_statistics`	GET	Query stats
`/system/accounts/{acct}/repositories/{repo}/query_history`	GET	Query history
`/system/accounts/{acct}/repositories/{repo}/revisions`	GET	Revisions
`/system/accounts/{acct}/repositories/{repo}/views/{view}`	GET/DELETE	View CRUD

5.4 History

Endpoint	Method	Purpose
`/system/service_history/transactions`	GET	Transaction history
`/system/service_history/imports`	GET	Import history
`/system/service_history/queries/{acct}/{sig}`	GET	Query text

5.5 Invitations

Endpoint	Method	Purpose
`/invitations`	GET	List invitations
`/invitations`	POST	Create/send invitation
`/invitations/{email}`	DELETE	Delete invitation

5.6 Graph Store Protocol

Endpoint	Method	Purpose
`/{acct}/{repo}`	GET	Export (content-negotiated)
`/{acct}/{repo}`	POST	Import (merge)
`/{acct}/{repo}`	PUT	Replace all
`/{acct}/{repo}`	DELETE	Clear
`/{acct}/{repo}/sparql`	POST	Execute SPARQL

---

6. File Structure

```
jsui/
  index.html                    # User SPA entry
```

```
admin.html                    # Admin SPA entry
app.js                        # User app bootstrap
admin-app.js                  # Admin app bootstrap
router.js                     # Shared client-side router
doc/
    requirements-20260205.md  # Current requirements
    summary.md                # This document
    analysis-20260204.md      # Comparative analysis
js/
    sparql-editor.js          # SPARQL editor (2,345 lines)
    yasqe-wrapper.js          # YASQE wrapper
lib/
    app_state.js              # Global state
    auth.js                   # Authentication
    auth_store.js             # Token storage
    config.js                 # Configuration
    models/                   # Data models
    persistence/              # Storage adapters
    replication/              # Change tracking
ui/
    app.js                    # User App controller
    routes.js                 # User routes (29)
    utils.js                  # Shared utilities
    components/               # Layout, Header, Footer, etc.
    pages/
        base_page.js          # Abstract base
        index.js              # User pages (~4,500 lines)
    admin/
        app.js                # Admin App controller
        layout.js             # Admin layout
        routes.js             # Admin routes (10)
        pages.js              # Admin pages (~1,550 lines)
stylesheets/
    style.css                 # Base styles
    jsui-overrides.css        # Custom overrides
images/
    Dydra Logo-user.svg       # User logo
    Dydra Logo-admin.svg      # Admin logo
    trash.svg                 # Delete icon
    link.svg                  # External link icon
fonts/, webfonts/             # Font assets
```

---

7. Technology Stack

- **Framework**: Vanilla ES6 modules, no build tooling
- **Editor**: YASQE (Yet Another SPARQL Query Editor)
- **HTTP**: Fetch API, XMLHttpRequest (for upload progress)
- **Storage**: LocalStorage for session persistence
- **Styling**: Static CSS (no preprocessor)
- **Icons**: SVG with PNG fallbacks