The History of Portal

A chronicle of 696 commits across two years, as told by the git log. One developer, one law firm, zero test suites.

Act I: The Late-Night Genesis (January - February 2024)

It starts at 3:20 AM on January 16, 2024. `first commit`. By 3:38 AM: `is a serve command enough?`. By 3:44 AM: `fix upload`. By 4:29 AM: `login page`. Then silence for two weeks.

On February 1st, the real work begins -- and immediately, the first battle. Six commits in a single hour fighting with NuxtLink:

```
does using NuxtLink istead of nuxt-link work?
remove explicit import
no explicit import of nuxtlink
test router link
testing links
try #2
try #3
try #3.5
finally
```

Five days later, the core of a case management system takes shape in a marathon session that stretches past 3 AM: searching, filtering, documents, services. The commit messages tell the emotional story: `fucking finally`, `I think the sql query is fixed now`, `fixed bad sql duplicating services`.

By mid-February, there's a real application: agencies, schools, representation emails, undo/redo, document uploads. The commit `reworked entire table whoohoo` on February 11 marks the first moment of genuine satisfaction. A version number appears: `v0.29.0` on February 27 -- suggesting 28 prior versions existed in some form before git entered the picture.

Sentry error tracking gets added. Changelogs start appearing. This is becoming real software.

Act II: The Agency Portal & Doc Generation (March 2024)

March brings hearing preparation features and the first attempt at multi-tenant architecture: an external agency portal. On March 10-11, a coding marathon runs from 6 PM to 4 AM -- `all the work so far`, `hearing edits`, `witness add/edit`, `agency portal`, `messing around with redirects`, `first pass at agency porta; ;` (the typo and double semicolon say everything about the hour).

The morning after: `remove redirects`. A false start.

URI encoding issues produce a string of increasingly exasperated commits: `sheehs`, `sheesh again`, `encode uri componenet`, `encode more`.

Document generation arrives by March 19 (`yay`), followed quickly by batch doc generation. The system can now generate legal documents at scale.

Act III: The Quiet Middle (April - September 2024)

A near two-month gap follows (late March to mid-May), broken only by a DocuSign webhook integration that requires four debugging commits before `this should fix it`.

June brings a schools API and the migration to type-safe queries with Kysely -- `typesafe queries`, `types and shit`, `no camel case for me:(`.

July sees bulk operations: bulk doc gen, bulk data export, bulk event lookup. But also frustration -- five consecutive commits just saying `dependency`, followed by a single-word commit: `fuck`.

August introduces a scheduler (`scheduler`, `scheduler take 2`). September brings Nitro tasks, calendar syncing, and Formsort webhook integration. The bones of an infrastructure are forming.

October through December 2024 is sparse. Small fixes. A quiet before the storm.

Act IV: The Puppeteer Crisis (February 9, 2025)

The most dramatic day in the repository's history.

It begins innocently on February 7 with `june 1 system` -- a new intake system that needs PDF generation. On February 9, the developer attempts to get Puppeteer and Chromium running on the server. What follows is 25+ commits in a single day, a slow descent into madness:

```
cache templates and satisfy weird ass requirements about commonJS imports
debug logs
use chromium executable
lets go hacky all the way
lets try this
remove the hacky way
lets see if this works
hmm
```

```
wtf
ok
will this work?
so all this was for nothing?
so all this was for nothing?
throwing shit at the wall to see what sticks
throwing shit at the wall to see what sticks
throwing shit at the wall to see what sticks
throwing shit at the wall to see what sticks
```

After four identical commits of despair, the session breaks. Hours later, a fresh approach:

```
chrome
more apt files
more apt files
lets see if npx works
lets see if npx works
postbuild?
programmatically?
fucking typescript
does installing it in pwd work?
bring back aptfile
bring back aptfile
bring back aptfile
lib this lib that
lets do this
```

And then: `log performance`. It works. `minor cosmetics`. `pupeteer page pool`. The crisis ends.

Three months later, in May 2025: `remove aptfile`, `remove puppeteer`. The Puppeteer approach is abandoned entirely in favor of something else. All of it was, in fact, for nothing.

Act V: Invoicing - The Feature That Never Ends (March - November 2025)

Invoicing development begins in March 2025 and doesn't truly finish for eight months. It arrives in waves:

Wave 1 (March-April): Basic invoicing, payment allocation. `invoicing`, `erros`, `payemtns and payment allocation`, `temp fix for invoicing`, `allocation`.

Wave 2 (September): A major push. `charges and all`, `more invoicing stuff`, `invoicing done` (narrator: it was not done). `invoicing changes`, `show hide invoiced`.

Wave 3 (October-November): The real invoicing sprint. In a single day (November 5), fourteen commits land: payment recording, statements, receipts, charge management, filters, search. `remove paypal` marks the end of an era.

Wave 4 (Late November): Invoice due dates, payment waterfall allocation, multi-year invoicing, auto-allocation, receipt sending history. On November 25 alone, twelve commits refine the invoicing system.

The invoicing system touches nearly every part of the application: charges, payments, statements, receipts, email sending, PDF generation, school year filtering, payor management.

Act VI: The Timezone Wars (Ongoing)

A recurring antagonist throughout the entire history. The timezone bug appears and reappears like a villain who refuses to stay dead:

- `fixed services dates timezone issue` (March 2024)
- `tz issue` (September 2024)
- `fix tiemzone issues` (April 2025)
- `tiemzone issues` / `timezone fix` (August 2025)
- `fixed timezone issue in analysis` (August 2025)
- `try fixing` / `try another fix for timezone issue in hearing table` (November 2025)
- `fxied school eyars` (October 2025)
- `fucking forgot to return dates as strings` (July 2025)
- `more date fucking fixes` (July 2025)
- `make dates dates again` (August 2025)

Every few months, dates break in a new and surprising way. The fix is always slightly different. The frustration is always the same.

Act VII: The AI Pivot (December 2025)

In late November 2025, something shifts. After `turn off ai for now` in October (a premature first attempt), AI comes roaring back in December.

The pace is extraordinary. In a single week (December 2-8): disclosure analysis with AI, DOCX export, exhibit categorization, streaming text generation, an IHO (Impartial Hearing Officer) dashboard, AI analysis for IHOs, IHO profile pages, PDF text extraction, rules extraction, and a migration from pg (node-postgres) to Kysely for the cases endpoint.

The second week brings IHO favorability scores, auto-scoring, roundtable email insights, and FOFD (Findings of Fact and Determination) tracking.

By mid-December, an SRO (State Review Officer) decision repository appears with batch processing. Then viability analysis tools, an RFR (Request for Review) drafter, full-text search with inline citations, DOCX export of research.

The legal research system expands beyond SROs. Model switching between Gemini and Claude arrives. Streaming research tools power AI subagents.

The developer works through Christmas Eve and Christmas Day: `Fix race condition in batch job progress tracking`, `Restructure IHO Data Repository`, `Redesign disclosure tool with zinc theme`, `Add model switching (Gemini/Claude) to legal research`.

By December 29, there's a full research sessions dashboard with permission-based management and real-time streaming via Redis/Valkey.

Act VIII: Production Hardening (January 2026)

The AI features are in production, and reality bites back.

Gmail sync breaks and needs monitoring. Timesheet generation has date bugs. The disclosure compilation produces bloated PDFs -- Ghostscript compression gets added. PDF encryption is detected and handled. Jobs get stuck and need recovery. Out-of-memory errors require limiting concurrent heavy jobs.

The commit messages shift from scrappy shorthand to structured descriptions: `Fix premature 'Failed' status during job retries`, `Limit concurrent heavy jobs to prevent OOM`, `Parallelize exhibit processing during compilation`, `Skip job queue startup plugin during build`.

The last commit (as of this writing): `Skip job queue startup plugin during build` on January 24, 2026. A build system fix. The unglamorous work of keeping production running.

By the Numbers

- **696 commits** over **2 years** (Jan 2024 - Jan 2026)
- **1 developer** (Abraham Neuwirth, sometimes committing as "Abe Neuwirth")
- **2 pull requests** in the entire history
- **0 test suites**
- **~12 timezone bugs** fixed (and counting)
- **25+ commits** in the Puppeteer Crisis alone
- **4 waves** of invoicing development
- Commits at **3 AM, 4 AM, 5 AM** are routine, not exceptional

The Branches That Tell Stories

- `feat-send-signature-*` (two attempts) -- signature sending needed multiple tries
- `fix-resend-invalid-retainer-*` -- retainer resending broke
- `abeneuwirth/ger-209-date-of-birth-when-edited-in-provider-portal-jumps-back-a` -- the branch name that got cut off, about a date-of-birth that jumps back a day. A timezone bug, obviously.
- `convert_immediately` -- the name suggests urgency

What the Commit Messages Reveal

The commit style evolves over two years. Early commits are raw and unfiltered: `haha`, `fuck`, `sheehs`, `yolo`, `cool`. Middle-period commits are terse: `fix`, `types`, `erros`. Late 2025 commits are structured and descriptive, likely AI-assisted: `Add PDF encryption detection and improved error messages`, `Fix premature 'Failed' status during job retries`.

The transition mirrors the application itself: from a scrappy internal tool hacked together in late-night sessions to a sophisticated legal technology platform with AI-powered research, real-time streaming, and production infrastructure.

Through it all, there's one developer, building an entire law firm's technology stack, one commit at a time.