# Developer Documentation

## Technical Architecture & System Overview

*Version 1.0 | 20th March 2026*

**Supported by**

netidee PROJEKTE

# Table of Contents

1. Project Overview
2. Architecture
3. Server Module
4. Editor Module
5. Player Module
6. Website Module
7. Shared Module
8. Data Flow & Integration
9. Technology Stack
10. Cloud Infrastructure
11. Development Setup

# 1. Project Overview

RichPods is a modern platform for creating and sharing enriched podcast content. It enables users to author podcast episodes enhanced with interactive multimedia chapters — including Markdown text, interactive charts, geographic maps, slideshows, polls, factboxes, and cards.

The platform is built as a pnpm monorepo with clear module boundaries. A GraphQL API server connects a Vue.js editor SPA, an embeddable player SPA, a Nuxt marketing website, and a shared component library.

## Key Capabilities

- Author rich, multimedia podcast chapters with diverse content types
- Embeddable player widget (similar to YouTube embeds)
- Podcast hosting with RSS feed generation
- Feed ownership verification system
- Multi-language support (English & German)
- Google Cloud Platform integration (Firestore, Cloud Storage, Cloud Run)

## Enclosure Types

Each RichPod chapter can contain one or more enclosures of these types:

| Enclosure Type | Description |
|---|---|
| Markdown | Rich text content rendered via TipTap (editor) and marked (player) |
| Interactive Chart | Data visualizations powered by ECharts |
| GeoMap | Geographic maps using MapLibre GL with GeoJSON data |
| Slideshow | Image galleries with swipe navigation |
| Poll | Interactive polls via the Coloeus service |
| Factbox | Highlighted information callouts |
| Card | 5 variants: Link, Cover, Citation, Image, Blank |

# 2. Architecture

The RichPods platform follows a modular monorepo architecture. Each module is independently buildable and deployable, but shares types, utilities, and assets through the shared package.

## Monorepo Structure

| Module | Purpose | Key Technologies |
|---|---|---|
| /server | GraphQL API | Express 5, Firestore, GCS |
| /editor | Authoring SPA | Vue 3, Tailwind, TipTap, Pinia |
| /player | Embeddable Player | Vue 3, ECharts, MapLibre GL |
| /website | Marketing Website | Nuxt 4, Vue 3 |
| /shared | Shared Library | Components, Utils, Assets, i18n |

## Module Interaction

All frontend modules communicate with the server exclusively via the GraphQL API. The server handles authentication, data persistence, file storage, and business logic.

**API Endpoint:** /graphql on port 4000 (development)

**Editor:** Port 5173 (Vite dev server)

**Player:** Port 5174 (Vite dev server)

**Website:** Port 3000 (Nuxt dev server)

# 3. Server Module

The server module provides the GraphQL API that all frontend applications consume. It is built on Express 5 with the graphql-http middleware and deployed on Google Cloud Run.

## Entry Point & Endpoints

| Route | Purpose |
|---|---|
| /graphql | GraphQL API endpoint |
| / | Ruru GraphQL Playground (dev) |
| /api/v1/upload | Image upload (multipart) |
| /api/v1/hosted | Hosted podcast management |
| /api/v1/og | Open Graph metadata extraction |

## Service Layer

The server implements 17 services organized by domain:

| Service | Responsibility |
|---|---|
| auth.service | Authentication & JWT verification |
| user.service | User profile management |
| richpod.service | RichPod CRUD & chapter management |
| podcast.service | Podcast metadata & episode search |
| feed.service | RSS/Atom feed parsing |
| storage.service | Google Cloud Storage operations |
| upload.service | Image upload processing (sharp) |
| verification.service | Feed ownership verification |
| hosted-podcast.service | Self-hosted podcast management |
| hosted-episode.service | Hosted episode management |
| hosted-storage.service | Hosted content storage |
| rss-feed.service | RSS feed generation |

| | |
|---|---|
| quota.service | Usage quota enforcement |
| rate-limit.service | API rate limiting |
| role-claims.service | Firebase custom claims management |

## Firestore Collections

| Collection | Purpose |
|---|---|
| users | User profiles and preferences |
| richpods | Core RichPod documents (with chapters sub-collection) |
| verifications | Feed ownership verification records |
| uploads | User-uploaded image metadata |
| hosted_podcasts | Self-hosted podcast configurations |
| hosted_episodes | Episodes within hosted podcasts |

## GraphQL Operations

### Queries

instanceInfo, richPod, userRichPods, recentPublishedRichPods, currentUser, user, podcastMetadata, extractFeedUrl, podcastEpisodeSearch, userVerifications, hostedPodcasts, hostedPodcast, hostedEpisodes, hostedEpisode

### Mutations

signUp, signIn, signInWithGoogle, updateProfile, createRichPod, updateRichPod, deleteRichPod, setRichPodChapters, startRichPodVerification, completeRichPodVerification, updateHostedPodcast, deleteHostedPodcast, deleteHostedEpisode

# 4. Editor Module

The editor is a Vue.js single-page application for authoring RichPods. It provides a rich editing experience with drag-and-drop chapter management, TipTap-based Markdown editing, and specialized editors for each enclosure type.

## UI Framework

Built with Tailwind CSS for styling, Headless UI for accessible components, and Ion Icons via the Iconify package. State management uses Pinia stores.

## Routes

| Route | Purpose |
|---|---|
| /signin | Sign-in / Sign-up page |
| /richpods | User’s RichPods list |
| /edit/:id? | RichPod editor workspace |
| /profile | User profile management |
| /verification | Feed ownership verification |
| /new-episode | Episode search |
| /hosted | Hosted podcasts list |
| /hosted/create | Create hosted podcast |
| /hosted/:id/edit | Edit hosted podcast |
| /hosted/:id/add-episode | Upload hosted episode |

## Key Stores (Pinia)

| Store | Responsibility |
|---|---|
| useRichPodStore | RichPod data, chapters, metadata, save state |
| useEditorUiStore | Validation errors, UI toggles |

## Key Composables

- useAuthState, useAuth, useCurrentUserRole
- useAutoSave, useSaveNow, useValidation
- useTipTapEditor, useUpload, useHostedUpload
- useColoeus, useQuota

## Enclosure Editors

| Editor Component | Implementation |
|---|---|
| MarkdownEditor | TipTap rich text editor |
| InteractiveChartEditor | ECharts with TinySpreadsheet |
| GeoMapEditor | TinyGeoJSON-based map editor |
| SlideshowEditor | Image gallery uploader |
| PollEditor | Coloeus poll builder |
| FactboxEditor | Styled Markdown callout |
| CardEditor | 5 card variants |

# 5. Player Module

The player is an embeddable Vue.js SPA designed to be embedded into any website, similar to a YouTube player. It renders published RichPods with full audio playback, chapter navigation, and interactive enclosure rendering.

## Audio System

| Composable | Function |
|---|---|
| useAudio() | HTML5 audio element lifecycle management |
| useMediaSession() | OS-level media controls (lock screen, headphones) |
| usePlaybackProgress() | localStorage persistence (up to 100 entries) |

## Enclosure Renderers

| Renderer | Technology |
|---|---|
| MarkdownEnclosure | marked + DOMPurify |
| InteractiveChartEnclosure | echarts-sandbox (iframe) |
| GeoMapEnclosure | MapLibre GL + GeoJSON |
| SlideshowEnclosure | @egjs/vue3-flicking |
| PollEnclosure | Coloeus (iframe) |
| FactboxEnclosure | Styled callout box |
| CardEnclosure | 5 card variants |
| UnsupportedEnclosure | Fallback for unknown types |

## Theming

The player uses CSS custom properties for full theming support. Key variables include colors, dimensions, font families, and gradient definitions. All theme tokens are defined in player/src/assets/theme.scss.

# 6. Website Module

The website is a Nuxt 4 application serving as the public-facing marketing site. It supports internationalization via @nuxtjs/i18n with German as the default language and prefix-except-default routing strategy.

## Configuration

| Setting | Value |
|---|---|
| Framework | Nuxt 4.3.0 |
| i18n | @nuxtjs/i18n 10.2.1 |
| Default Locale | de (German) |
| Routing Strategy | prefix_except_default |
| GraphQL Endpoint | Configurable via runtime config |

# 7. Shared Module

The shared module provides reusable components, utilities, i18n resources, and assets consumed by all frontend modules. It is published as @richpods/shared within the monorepo.

## Exports

| Import Path | Contents |
|---|---|
| @richpods/shared/i18n/language | Language detection and helpers |
| @richpods/shared/utils/roles | User role definitions |
| @richpods/shared/utils/itunesCategories | iTunes podcast categories |
| @richpods/shared/assets/* | SVG logos, fonts, images |

## i18n Strategy

Shared locale files (en.json, de.json) are merged into the editor and player i18n setups. The website uses its own locale catalog via @nuxtjs/i18n.

# 8. Data Flow & Integration

## Authentication Flow

1. Editor uses Firebase Web SDK (email/password or Google OAuth)
2. Firebase issues ID tokens (JWT)
3. Editor injects JWT as Bearer token in GraphQL requests
4. Server verifies token via Firebase Admin SDK
5. Custom claims assign user roles (super_admin, editor)

## Content Creation Flow

1. Editor collects user input, validates locally with Zod
2. GraphQL mutations send data to the server
3. Server validates with Joi, stores in Firestore
4. Enclosures and images are uploaded to Cloud Storage
5. Player queries published RichPods via GraphQL
6. Renderers display each enclosure type interactively

## GraphQL Code Generation

All modules use graphql-codegen to generate TypeScript types from the server schema. The server generates resolver types, while editor and player generate typed SDK functions.

## 9. Technology Stack

| Technology | Version | Usage |
|---|---|---|
| Node.js | 24.0.0+ | Runtime |
| pnpm | 10.30.0 | Package manager |
| Vue.js | 3.5.x | Frontend framework |
| TypeScript | Catalog | Type system |
| Vite | 7.x | Build tool |
| Express | 5.1.0 | Server framework |
| GraphQL | Catalog | API layer |
| Firestore | 7.11.6 | Database |
| Firebase Admin | 13.6.0 | Server auth |
| Tailwind CSS | 3.4.19 | Editor styling |
| TipTap | 3.20 | Rich text editor |
| ECharts | 6.0 | Charts |
| MapLibre GL | Catalog | Maps |
| Pinia | 3.0.4 | State management |
| Nuxt | 4.3.0 | Website framework |
| vue-i18n | Catalog | Internationalization |

# 10. Cloud Infrastructure

## Google Cloud Services

| Service | Usage |
|---|---|
| Cloud Firestore | Primary document database (6 collections, 18 composite indexes) |
| Cloud Storage | 3 buckets: enclosures, uploads, hosted content |
| Firebase Auth | User identity, sign-in, JWT tokens, custom roles |
| Cloud Run | Server deployment (1 vCPU, 256 MB, autoscaling 0–1) |
| Cloud Functions | validate-mp3 (storage-triggered), check-verifications (HTTP) |
| Secret Manager | Postmark API token and other secrets |
| Artifact Registry | Docker images for Cloud Run |
| Workload Identity | Keyless CI/CD auth from GitHub Actions |

## Storage Design

| Bucket | Object Path Pattern |
|---|---|
| Enclosures | {richpodId}/{timestamp}-{type}-{uuid}.json |
| Uploads | {richpodId}/{uuid}.{extension} |
| Hosted Audio | {podcastId}/{episodeId}/{filename} |
| Hosted Covers | {podcastId}/channel/{filename} |

**Caching:** All storage objects are cached for 1 year (immutable, public)

## CI/CD Workflows

| Workflow | Deployment Target |
|---|---|
| deploy_server.yml | Cloud Run (Docker + Artifact Registry) |
| deploy-player.yml | Remote server (Vite build + rsync SSH) |
| deploy-editor.yml | Remote server (Vite build + rsync SSH) |
| deploy-website.yml | Remote server (Nuxt generate + rsync SSH) |
| deploy-validate-mp3.yml | Cloud Functions (Gen2) |
| deploy-check-verifications.yml | Cloud Functions (Gen2) |

# 11. Development Setup

## Prerequisites

- Node.js 24.0.0+ (see .nvmrc)
- pnpm 10.30.0+
- Google Cloud credentials for Firestore/GCS access

## Development Commands

| Command | Description |
|---|---|
| pnpm dev:server | Start server on :4000 |
| pnpm dev:editor | Start editor on :5173 |
| pnpm dev:player | Start player on :5174 |
| pnpm dev:website | Start website on :3000 |
| pnpm build:<module> | Build any module |
| pnpm test:<module> | Run tests |
| pnpm lint | Lint all modules |
| pnpm format | Format code with Prettier |

## Code Conventions

- TypeScript: "any" type prohibited, prefer type over interface
- Vue: <script setup lang="ts">, PascalCase components, use-prefix composables
- Styling: SCSS scoped by default, Tailwind in editor only
- Formatting: Prettier with 4-space indent, double quotes, semicolons, 100 char width
- Functional & declarative patterns — no classes
- Hard cutover approach, no backward compatibility