START

- Overview
- Getting started
- Languages

DISCOVER MUSIC

- Home
- Rich search
- People & ensembles
- Questions
- Work page
- Performance page
- Performances
- By scoring
- Instruments over time
- Timeline & map
- Networks
- Recommendation
- Text generation
- RDF data

MUSIC CATALOGUE

- Dashboard
- Editing a record
- Works
- Events
- Programming
- Recordings
- Post-production
- Publications
- Agents
- Vocabularies
- From a concert programme
- DOREMUS model

REFERENCE

- Data and sources
- Administration
- Model codes
- Technical architecture

USER GUIDE

Cataloguing and discovering music with the DOREMUS model

The application has two spaces built on the same knowledge graph: **Discover music**, to search and explore works, concerts, performers and recordings, and the **Music catalogue**, to describe them following the DOREMUS ontology. This guide walks through every screen.

Overview

Discover music

Read-only, for the public and for librarians: rich search, people, questions in plain language answered by Claude, performance history, scoring, instruments over time, timeline and map, networks of performers, explained recommendations and generated programme notes.

Music catalogue

For cataloguers: records of works, events (planned and actual), programming, recordings, post-production, publications, persons and corporate bodies, each one checked against the model's rules (SHACL) before it is saved, with its RDF visible at all times.

Both spaces read the same triplestore (Apache Jena Fuseki). It holds a real extract of data.doremus.org — BnF, Philharmonie de Paris and Radio France data — and the records created in the catalogue. The DOREMUS model extends FRBRoo and CIDOC-CRM: a work (F14) is realised by an expression (F22), a concert is a performance (F31), an announced concert is a foreseen performance (M26), and so on (see Model codes).

A record of data.doremus.org is never changed. When you open one in the catalogue to enrich it, the application copies it into a new catalogue record that points to the original (`owl:sameAs`, `prov:wasDerivedFrom`); from then on Discover shows the enriched record instead of the original.

Getting started

Open the application in a browser at the address of the server (on a local installation: `http://localhost:8000`). The start page leads to the two spaces:

- **Discover music**: `/explore.html`
- **Music catalogue**: `/app.html`
- **This guide**: `/guide/index.html`, also linked from both spaces (« Documentation »).

Each space has a link to the other in its top bar (« ↔ Cataloguing space », « ↔ Discover »). In Discover, the **model codes** box shows the classes and properties of the model beside each heading (F22, M6 → M23…), for those who want to see how a page is built on the graph.

Languages

The interface is available in French, English and German: use **FR · EN · DE** at the top right of either space. The choice is remembered by the browser. A link can also give the language: `/explore.html?lang=de`.

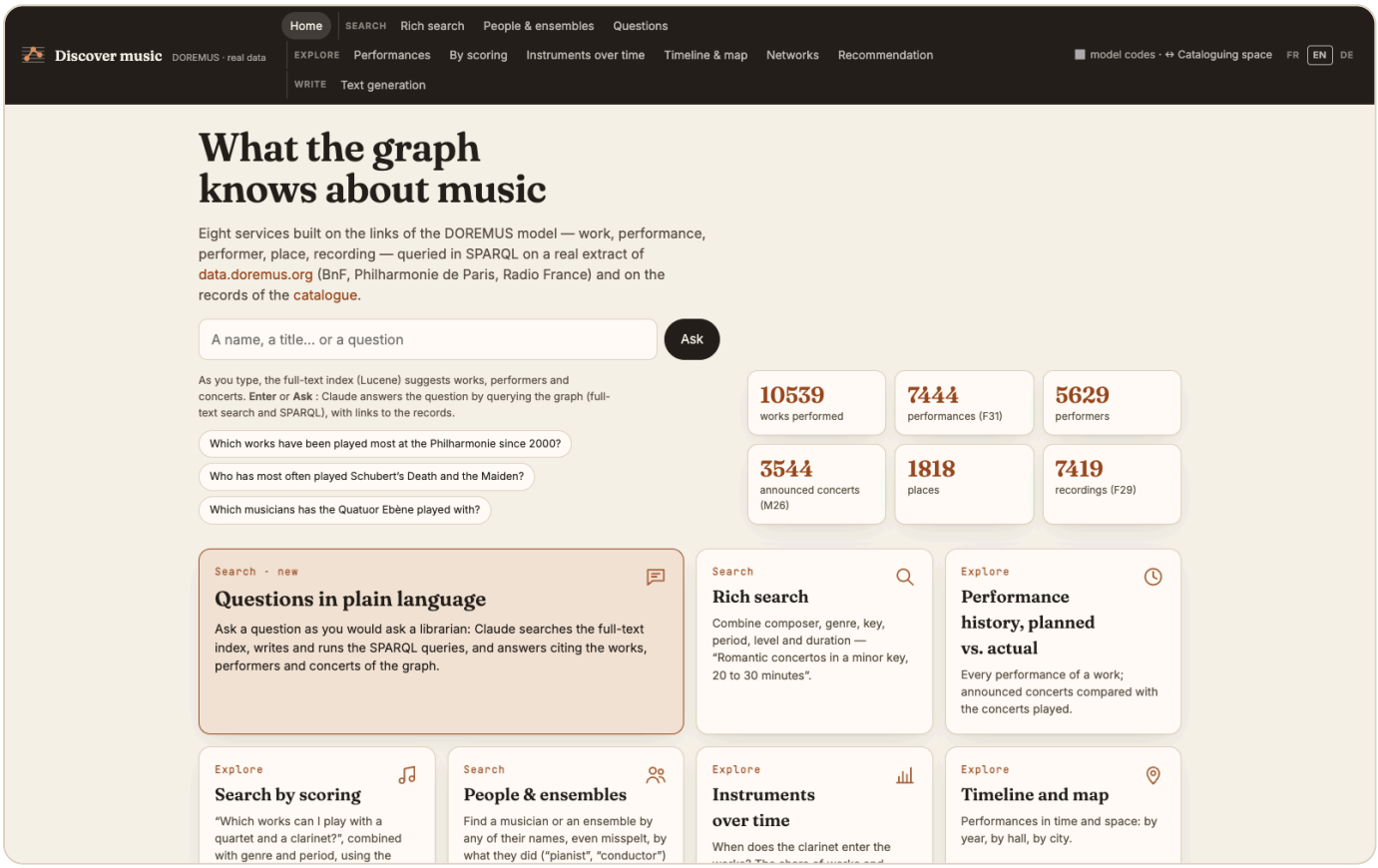
- Menus, headings, buttons, help texts and messages are translated.
- Concept labels (keys, instruments, genres, functions…) come from the vocabularies in the chosen language (« D Minor », « D Moll »). The IAML vocabularies have no German labels: German shows the English label.
- Programme notes are written by Claude in the interface language, and the chat answers in the language of the question.
- The data stays as the sources give it: work titles (« Symphonies. Ré mineur. No 9 »), names, notes and biographies.

Discover music

The menu groups the services in three families: **Search** (rich search, people and ensembles, questions), **Explore** (performances, scoring, instruments over time, timeline and map, networks, recommendation) and **Write** (text generation). Every page title, work, performer or concert is a link; every page has a link to its RDF data.

Home

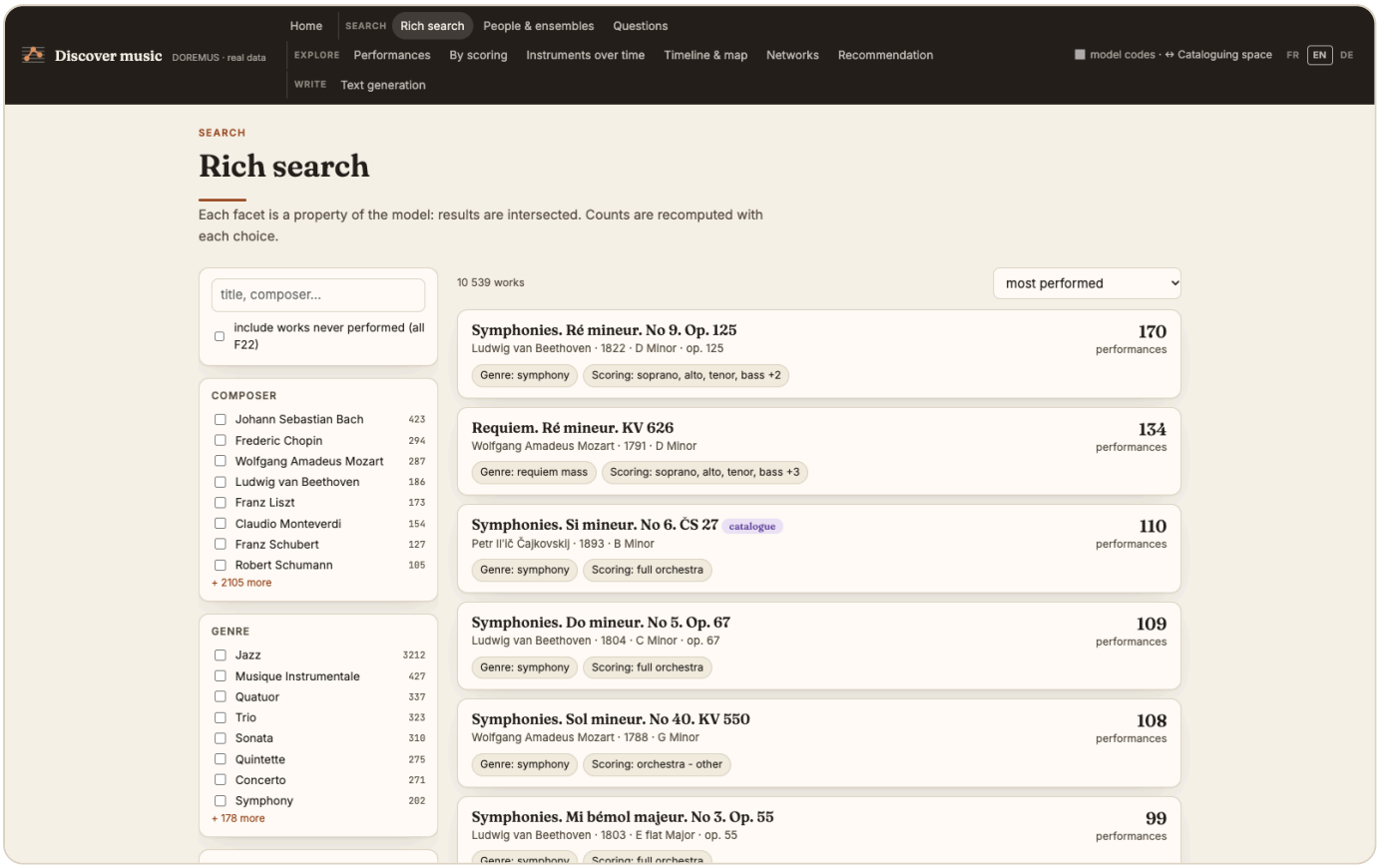
The home page has a single search box: type a name or a title and the full-text index suggests works, performers and concerts as you type; type a question and press Ask to send it to Claude. Below, the figures of the corpus and one card per service.



Home page: search box, sample questions, figures of the corpus and the services.

Rich search

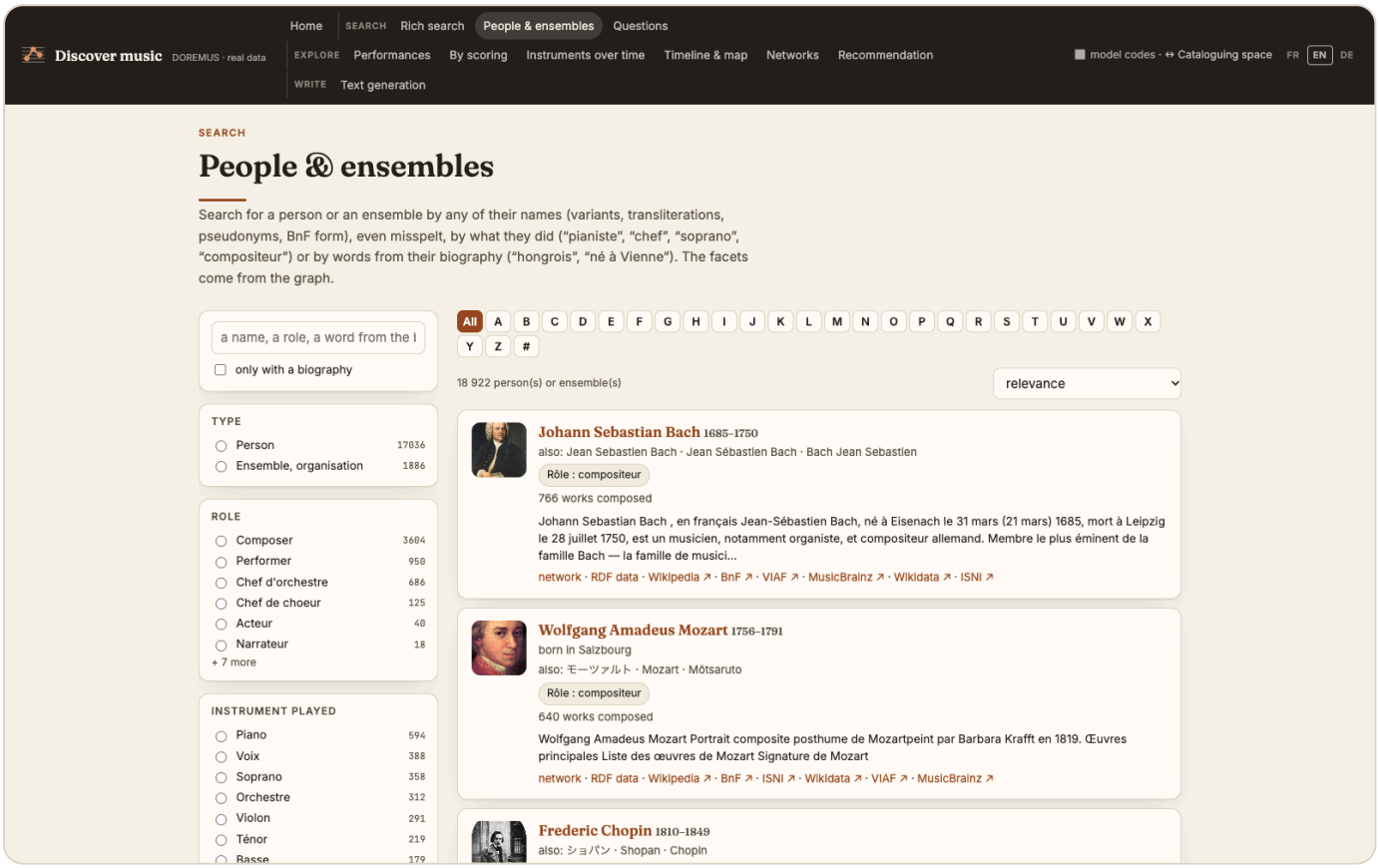
Combine facets that are properties of the model: composer, genre, period, key, instrument family, level and duration. Results are the intersection of the choices, and the counts beside each value are recomputed after every click. Sort by number of performances, creation date, duration or title. By default only works performed in a catalogued concert or recorded are listed; tick **include works never performed** for all expressions (F22).



Rich search: facets on the left, works with their genre and scoring on the right.

People & ensembles

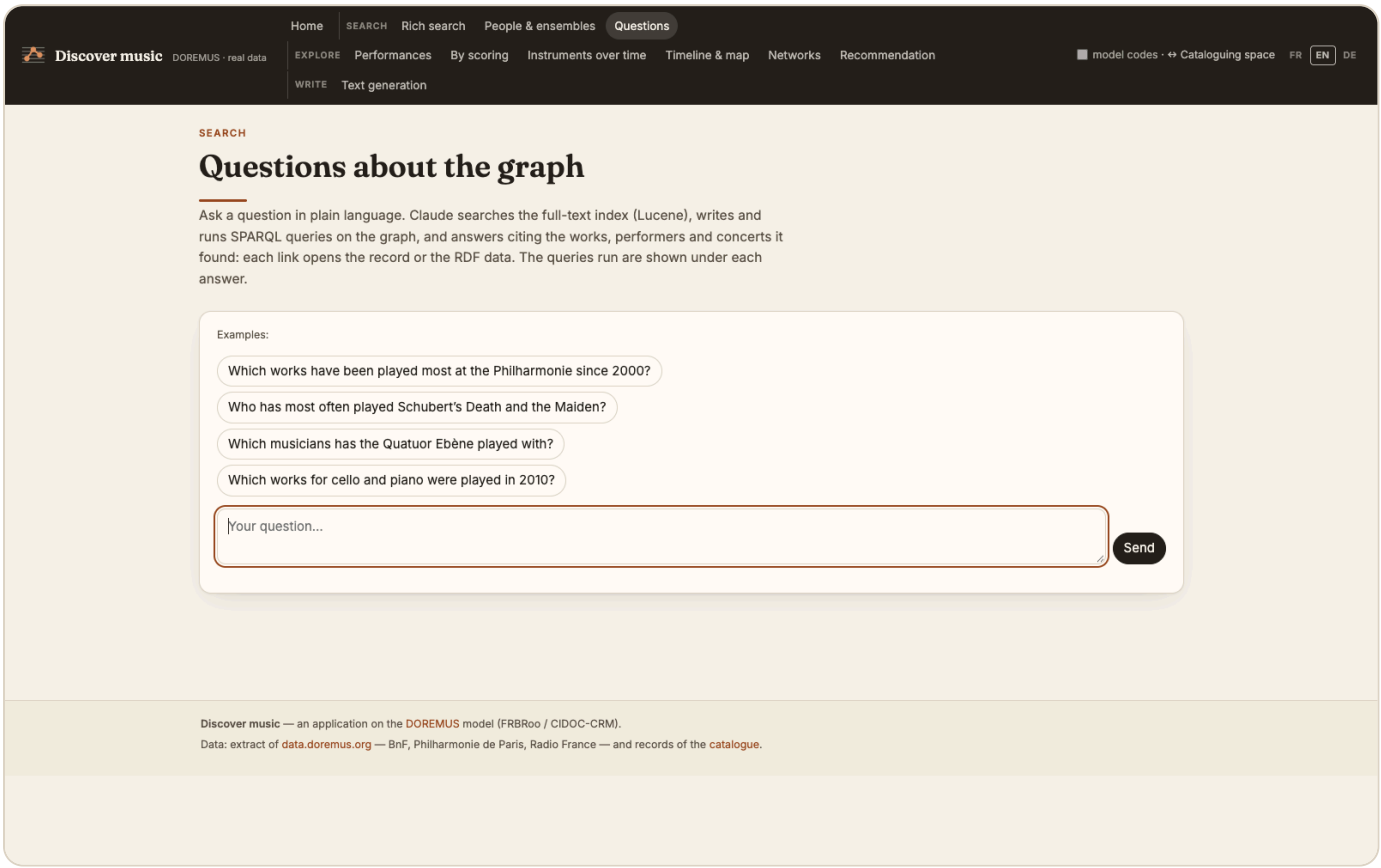
Find a person or an ensemble by any of their names — variants, transliterations, pseudonyms, the BnF form — even misspelt (« bernstien »), by what they did (« pianiste », « chef ») or by words of their biography (« né à Vienne »). Facets: type, role, instrument played, century and place of birth, gender; an A–Z index browses the list by surname. Each row links to the person's network, their RDF data and their Wikipedia page when the alignments found one.



People & ensembles: search box, facets, alphabetical index and results with life dates and biography.

Questions

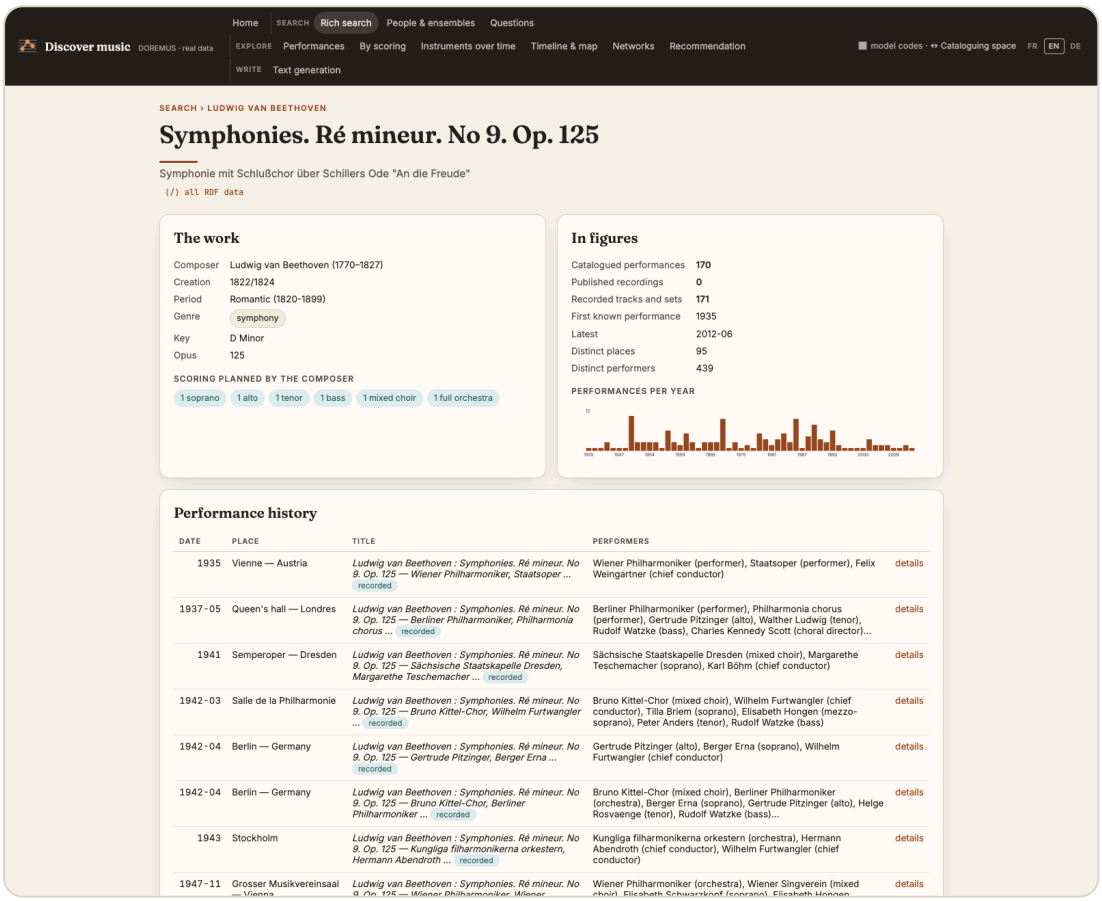
Ask a question in plain language, as you would ask a librarian: « Who has most often played Schubert's Death and the Maiden? ». Claude searches the full-text index, writes and runs SPARQL queries on the graph, and answers with links to the works, performers and concerts it found. The queries it ran are shown under each answer. Answers rely only on the graph: « no result » means « not in the extract », not « never happened ». This service needs an Anthropic API key on the server.



Questions: type a question, Claude answers from the graph and shows its queries.

Work page

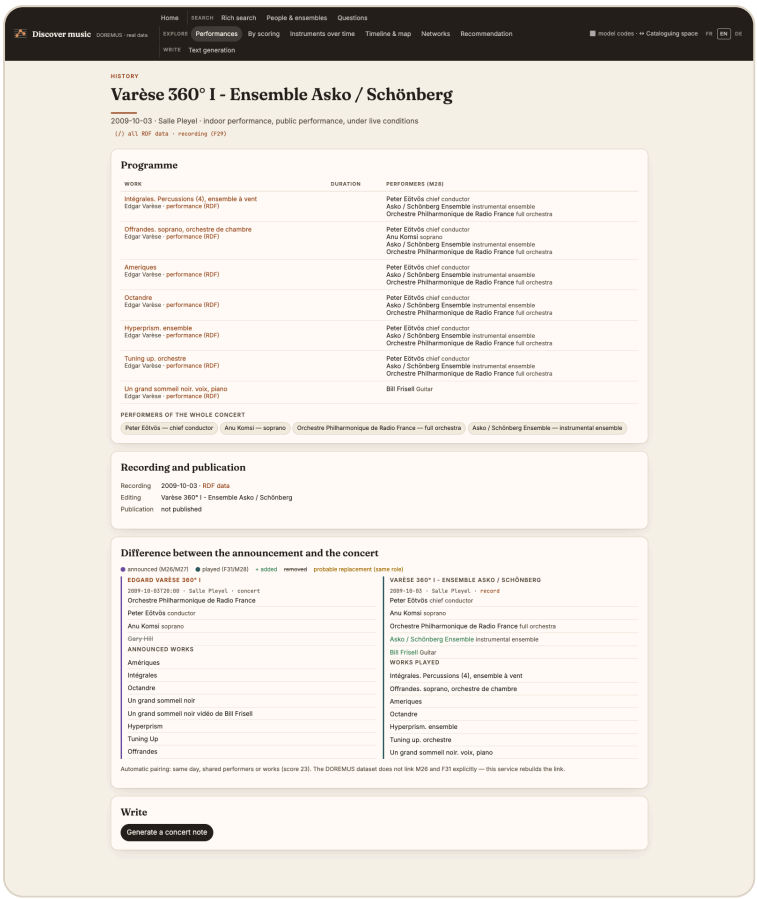
The page of a work gathers what the graph knows: composer, creation, genre, key, opus, duration and the scoring planned by the composer (M6 → M23); its figures (catalogued performances, published recordings, first and last performance, places, performers) and a chart of performances per year; the full history of its performances; its **recordings** (tracks and sets of tracks, from the radio, the studio or a recorded concert, with or without a catalogued concert); related works and a button to generate a programme note.



Work page: the work, its figures, the chart of performances per year, then the history and the recordings.

Performance page

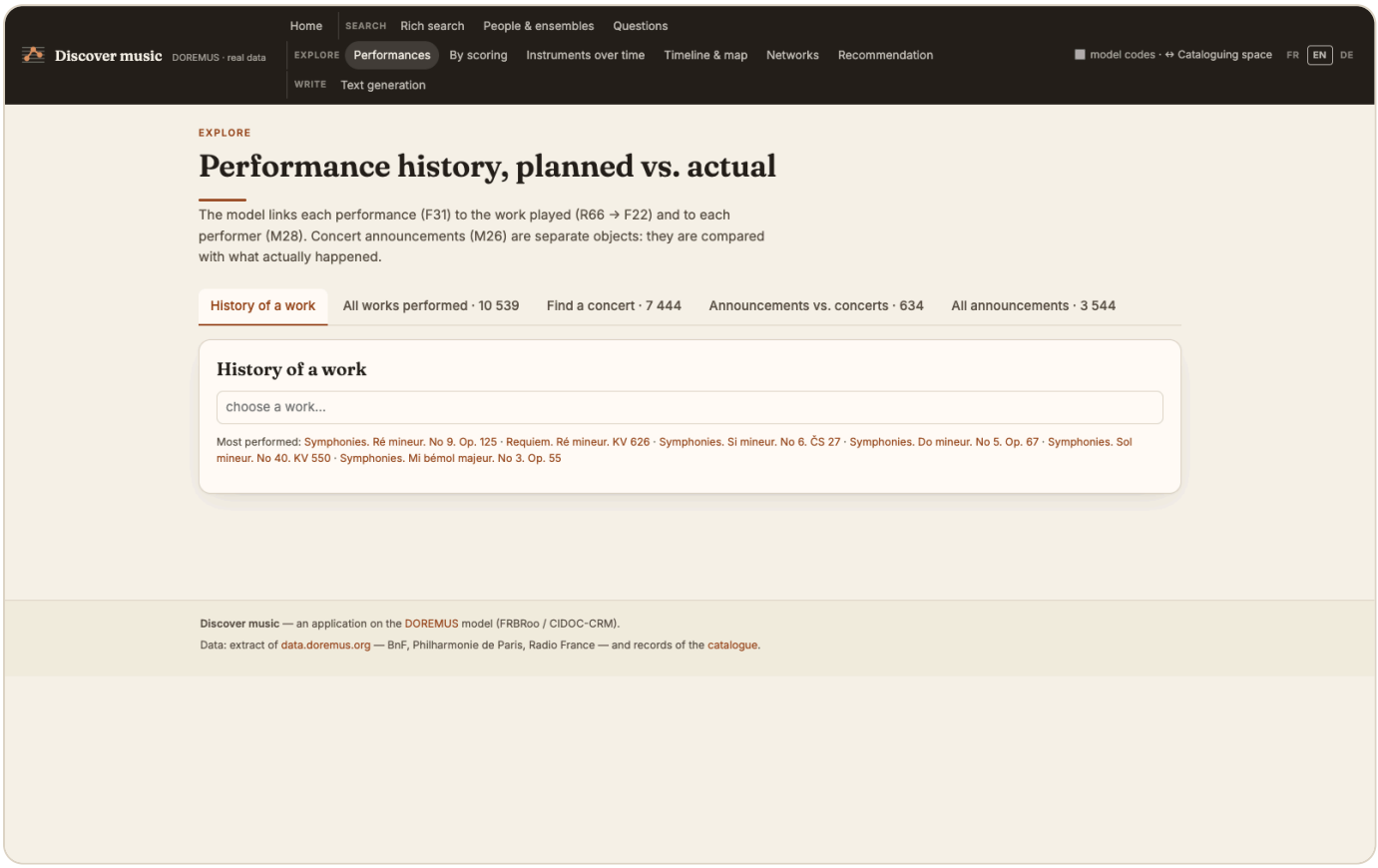
A concert (F31): its programme with the performers of each work and their role, the performers of the whole concert, the recording and publication chain when there is one, and — when an announcement (M26) was paired with it — the difference between what was announced and what was played: replacements, additions, removals.



Performance page: programme, performers and the comparison with the announced concert.

Performances

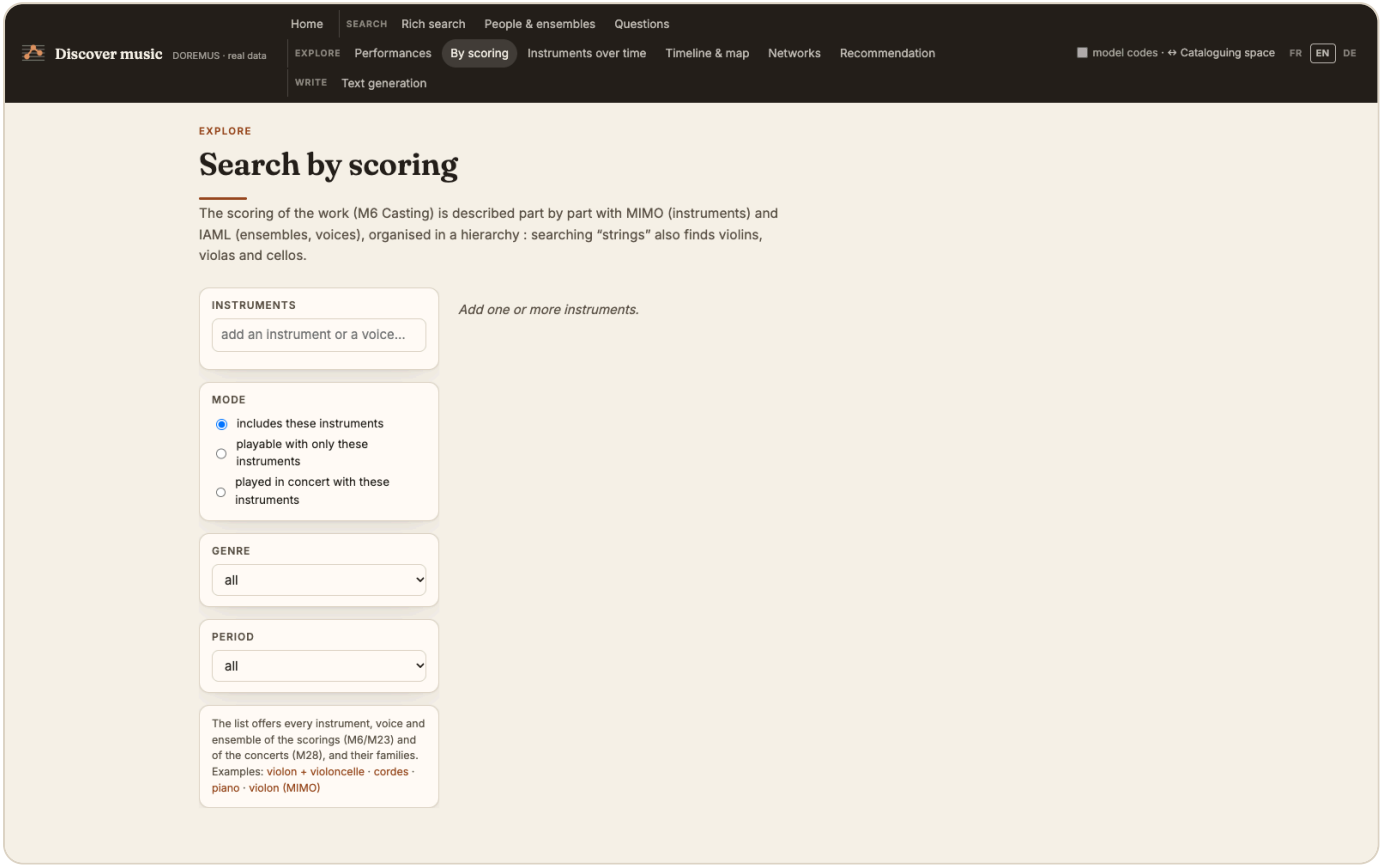
Four sections, reached by the tabs at the top of the page: the **history of a work** (choose a work, see all its performances), the list of **all works performed**, a **concert search** (title, date such as `2008-10`, hall, city, performer, work, composer) and the **announcements compared with the concerts**. The data does not link an announcement (M26) to its concert (F31): the application pairs them (same day, then shared performers and works).



Performances: history of a work, concert search, and announcements compared with the concerts played.

By scoring

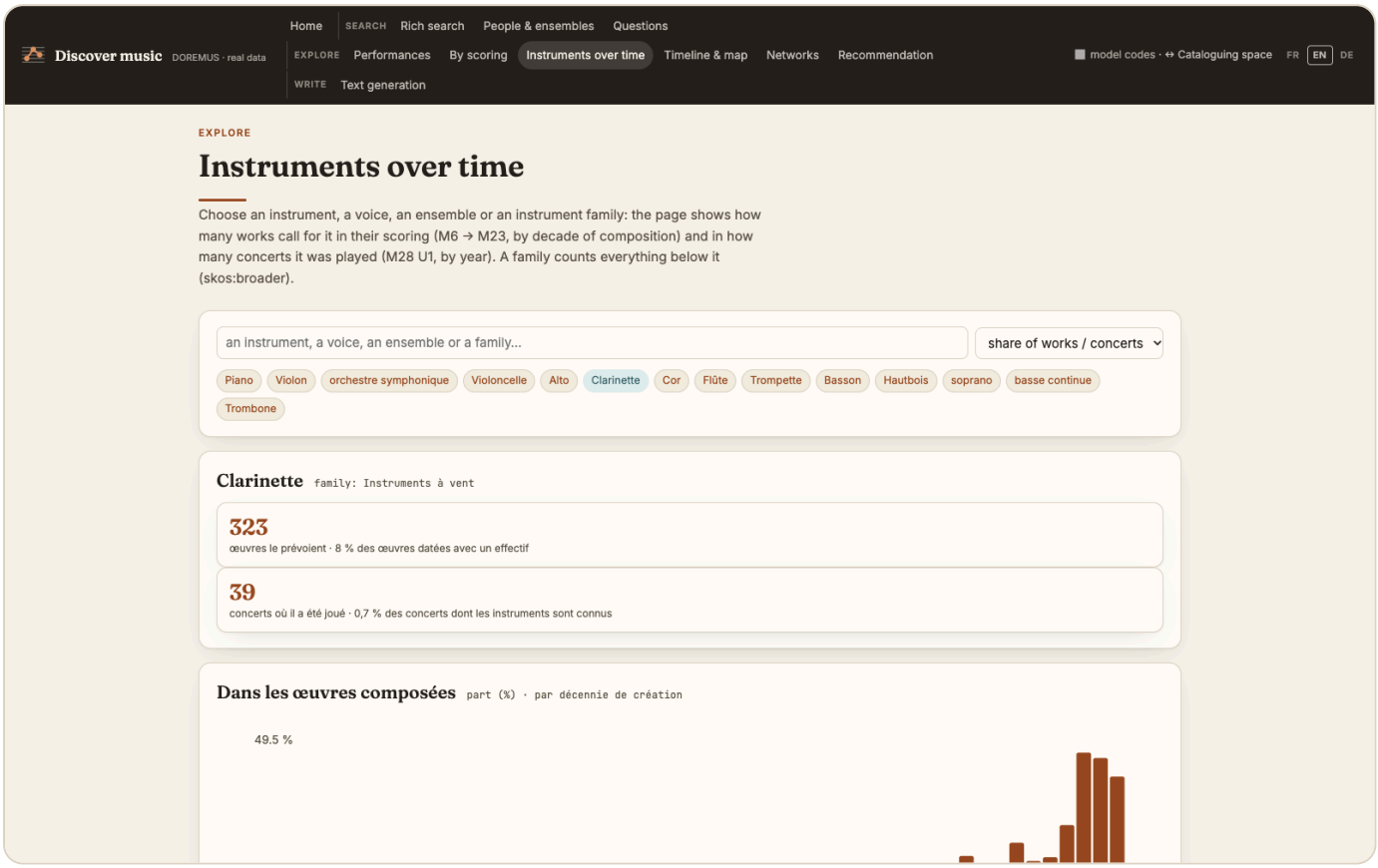
« Which works can I play with a quartet and a clarinet? ». Add instruments, voices or ensembles; choose whether the works must *include* them, be *playable with only* them, or have been *played in concert* with them; narrow by genre and period. The instrument hierarchy is used: « strings » also finds violins, violas and cellos.



By scoring: instruments, mode and examples.

Instruments over time

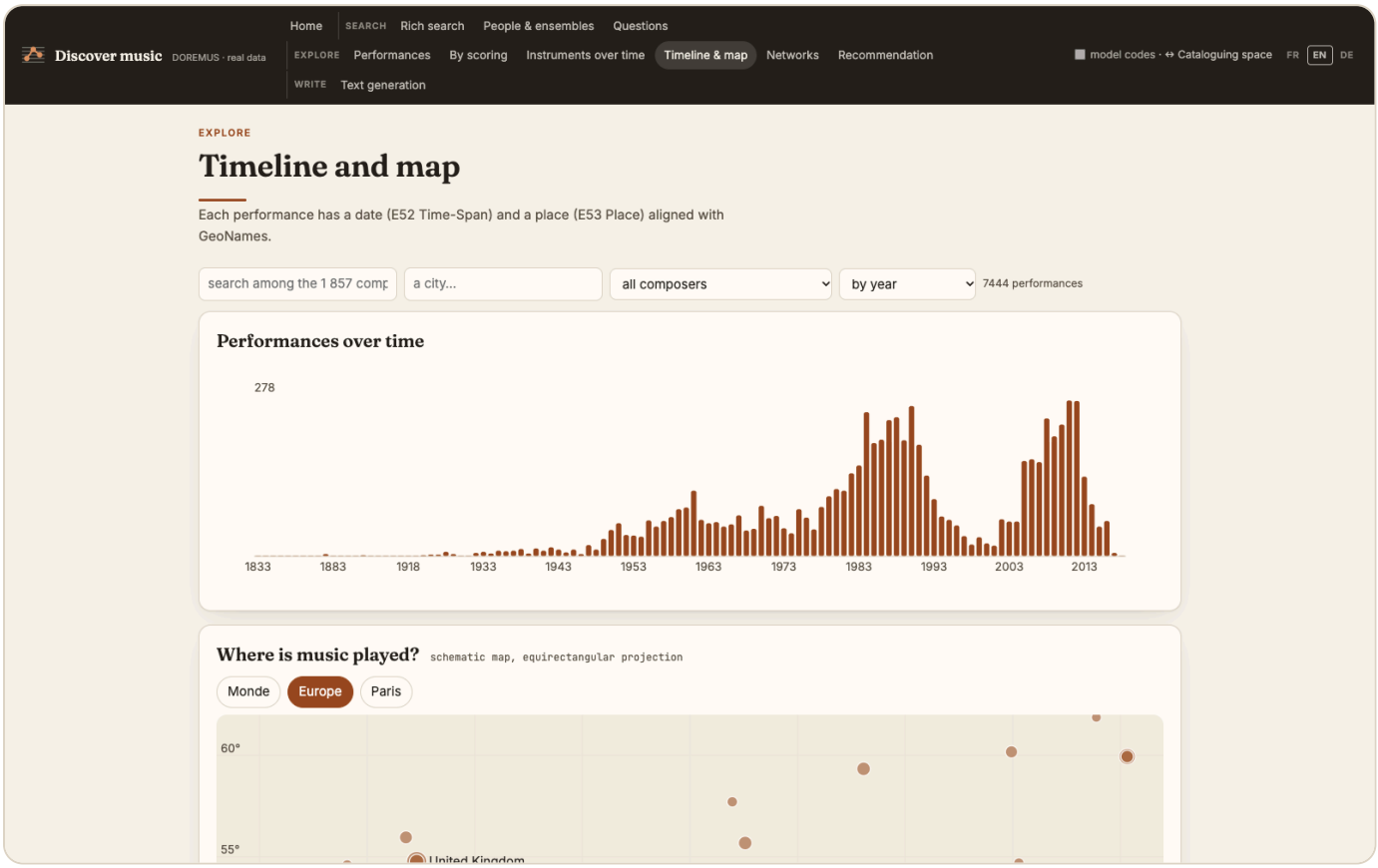
Choose an instrument, a voice, an ensemble or a family: the page shows how many works call for it in their scoring, by decade of composition, and in how many concerts it was played, by year — as a number or as a share of all works and concerts. A family counts everything below it.



Instruments over time: the clarinet in the works and in the concerts.

Timeline & map

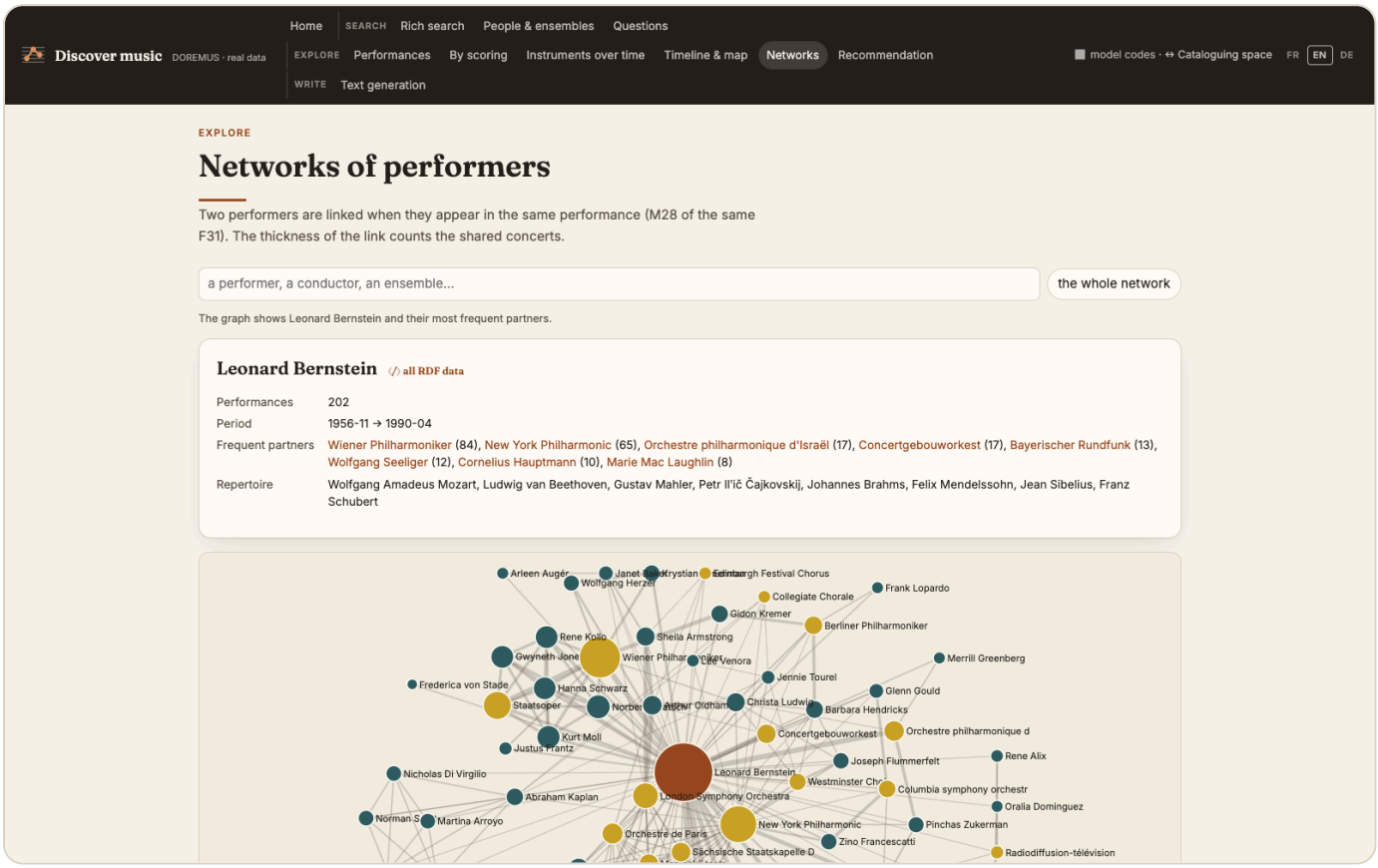
Performances in time and space: a chart by year or by decade, filtered by composer or city; a map of the places (coordinates from GeoNames; the size of a circle is the number of performances); the cities and their halls.



Timeline & map: performances over time, map of the places and list of cities.

Networks

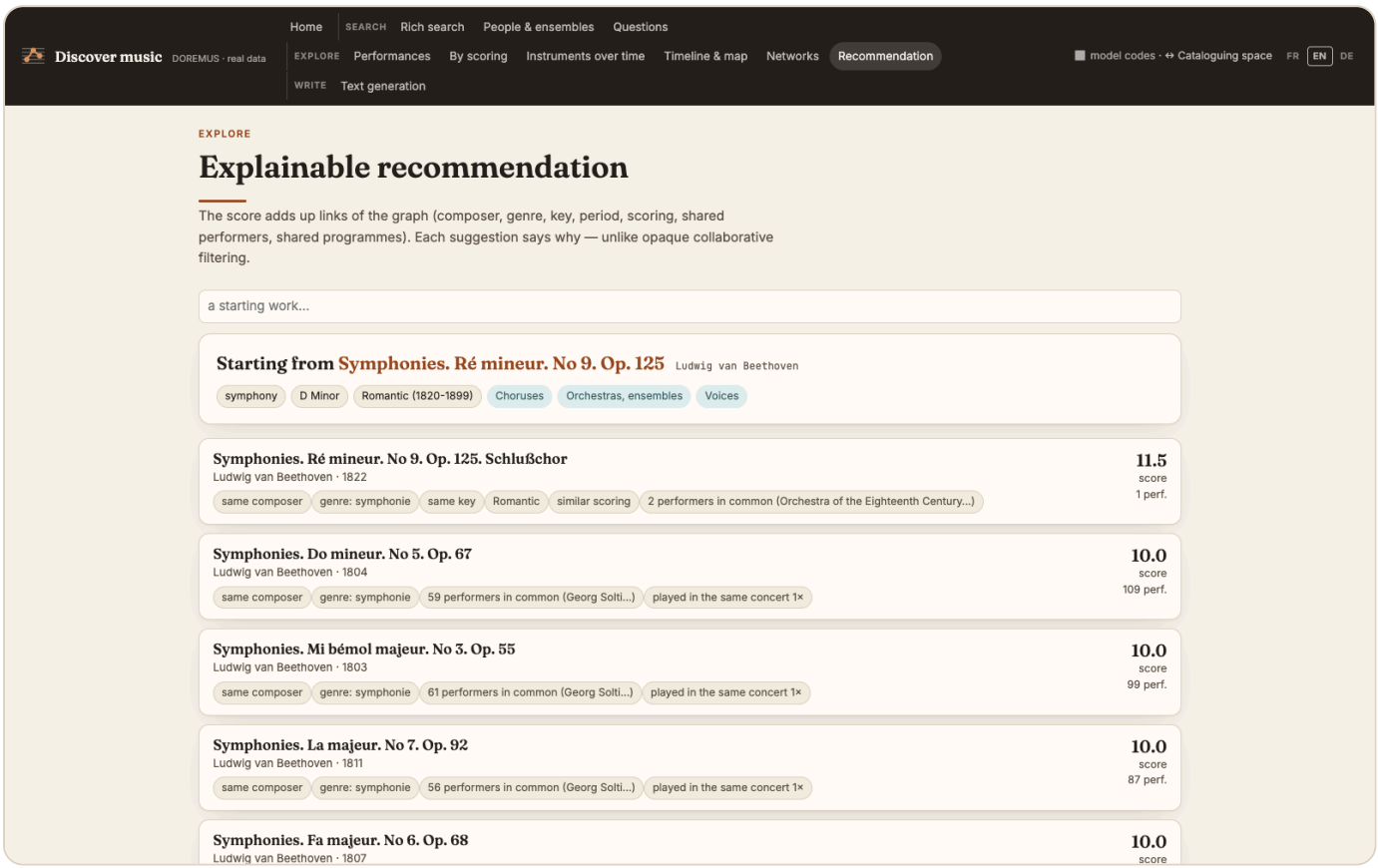
Who plays with whom: two performers are linked when they appear in the same concert, and the thickness of the link counts the concerts they shared. Click a node to centre the network on it; the list below gives every performer with their main roles, period and partners.



Networks: Leonard Bernstein and his most frequent partners.

Recommendation

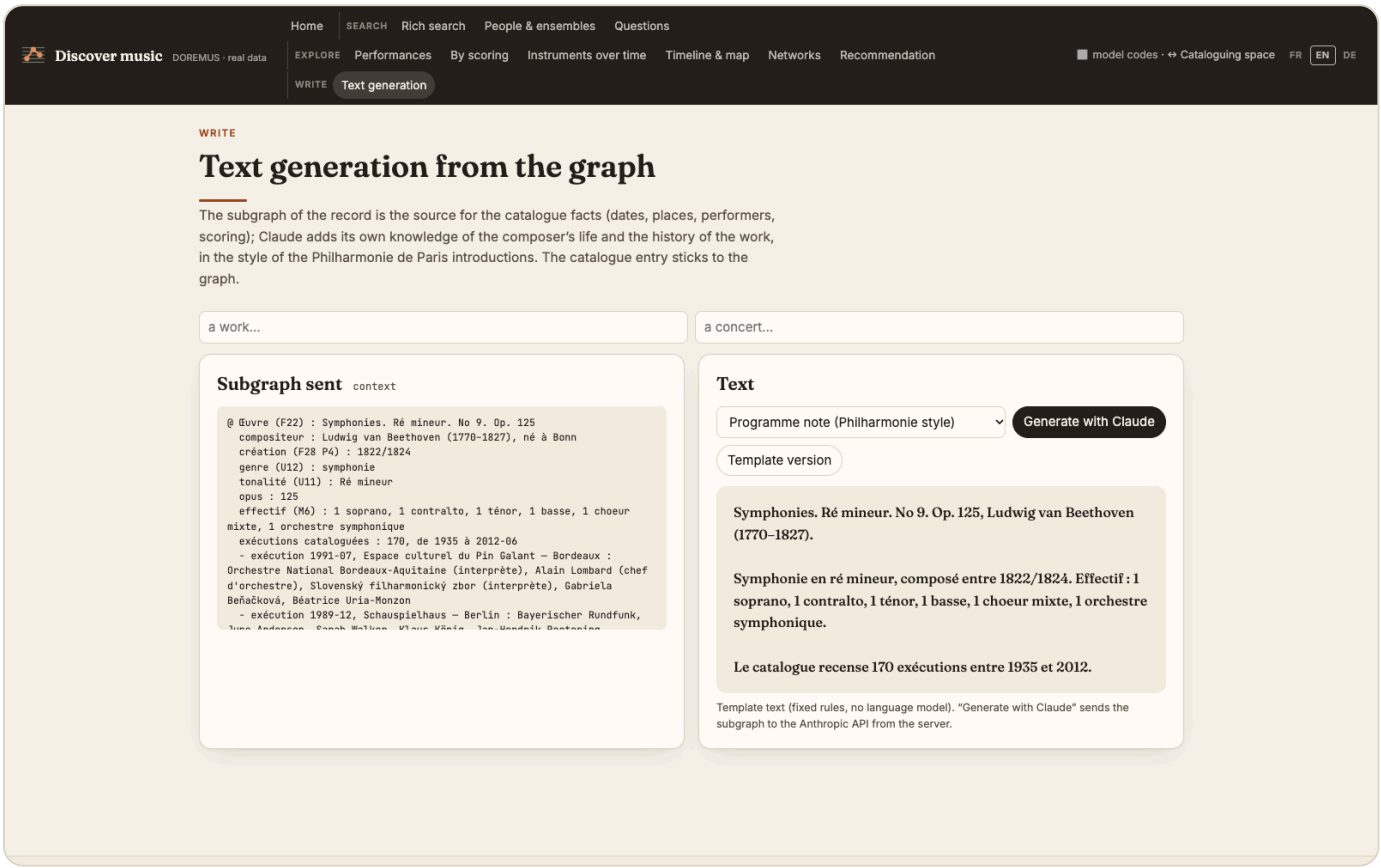
Related works scored on the links of the graph — same composer, genre, key, period, similar scoring, shared performers, played in the same concert — and each suggestion says why.



Recommendation: related works with the reasons for each one.

Text generation

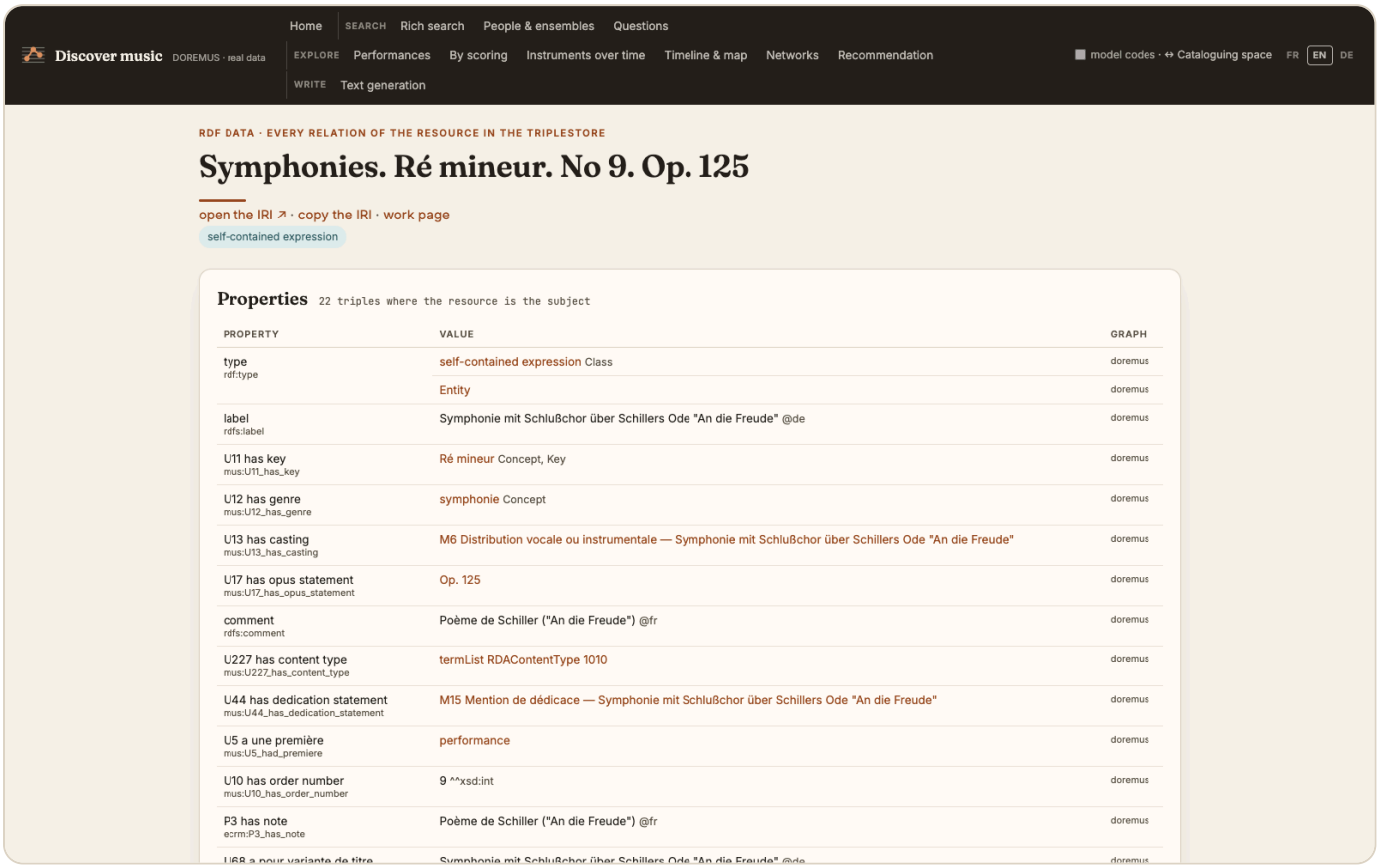
Choose a work or a concert and a style — concert introduction, programme note, catalogue entry, social media post. The subgraph of the record (shown on the page) gives the catalogue facts; for the introduction, the programme note and the post, Claude adds its own knowledge of the composer and the work, in the style of the Philharmonie de Paris presentations. The catalogue entry sticks to the graph. Without an API key, a template text built with fixed rules is shown.



Text generation: the subgraph sent to Claude, the styles and the text.

RDF data

Every resource has a page with all its triples: the properties where it is the subject, the incoming relations where it is the object, and the graph each triple comes from. Each value is a link to its own RDF page.



RDF data: properties and incoming relations of a work.

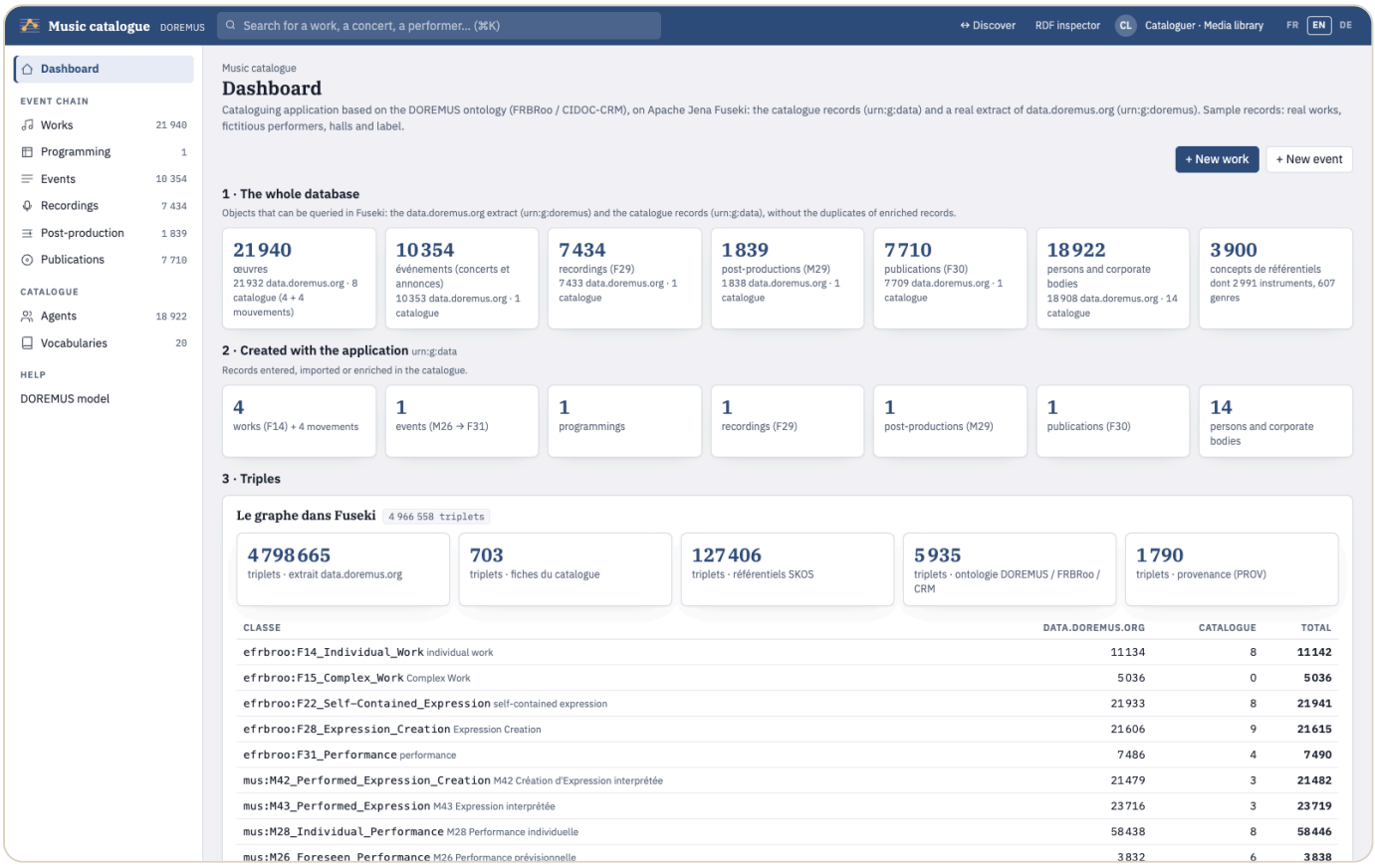
MUSIC CATALOGUE

Describing music in the catalogue

The left menu follows the chain of the musical event — works, programming, events, recordings, post-production, publications — then the agents and the vocabularies. The search box at the top (⌘K) finds a work, a concert or a performer. Each list shows the catalogue records and, below, the resources of data.doremus.org of the same kind.

Dashboard

Counts for the whole database and for what was created with the application, the chain of the musical event (each step is an FRBRoo triad: an event creates a work and its expression; what is planned is distinct from what happened) and a list of things to do.



Dashboard of the catalogue.

Editing a record

Every record has the same buttons at the top right:

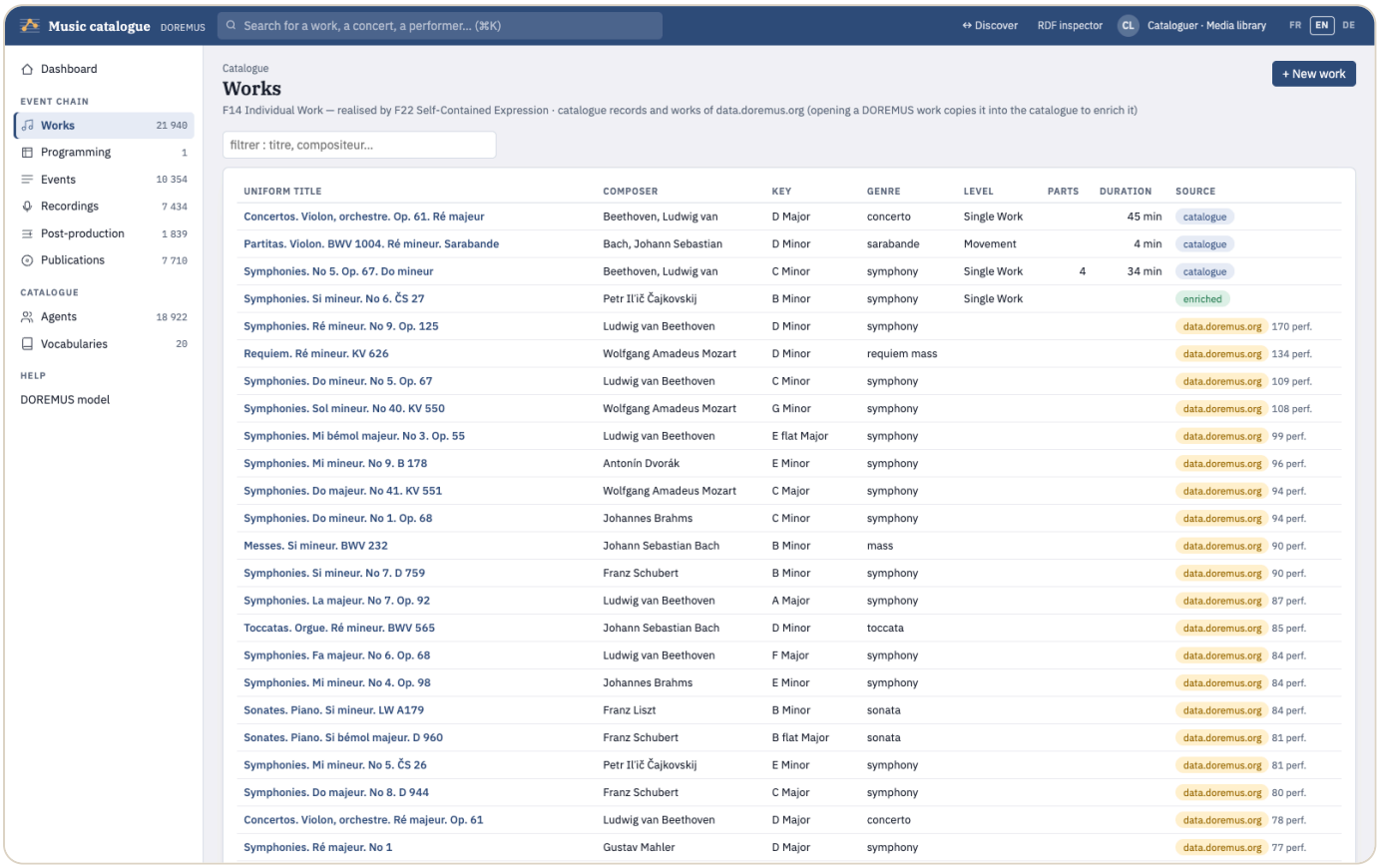
BUTTON	WHAT IT DOES
Check the RDF	Checks the record against the rules of the DOREMUS model (SHACL shapes) and shows the report and the RDF on the right. Nothing is saved.
Discard	Discards the unsaved changes: the record goes back to its last saved version. A record never saved (new, or copied from data.doremus.org) is dropped; the original resource is unchanged.
Save	Checks the record, then saves it in the catalogue graph. Refused while errors remain; warnings do not prevent saving.

Beside the buttons, a status says whether the record is **saved**, has **unsaved changes**, or is a **new record** (or a copy of data.doremus.org) not saved yet. Leaving the page with unsaved changes asks for confirmation. The **RDF inspector** (top bar) shows, at all times, the RDF graph of the record as the API generates it, and the SHACL report: errors in red, warnings in orange.

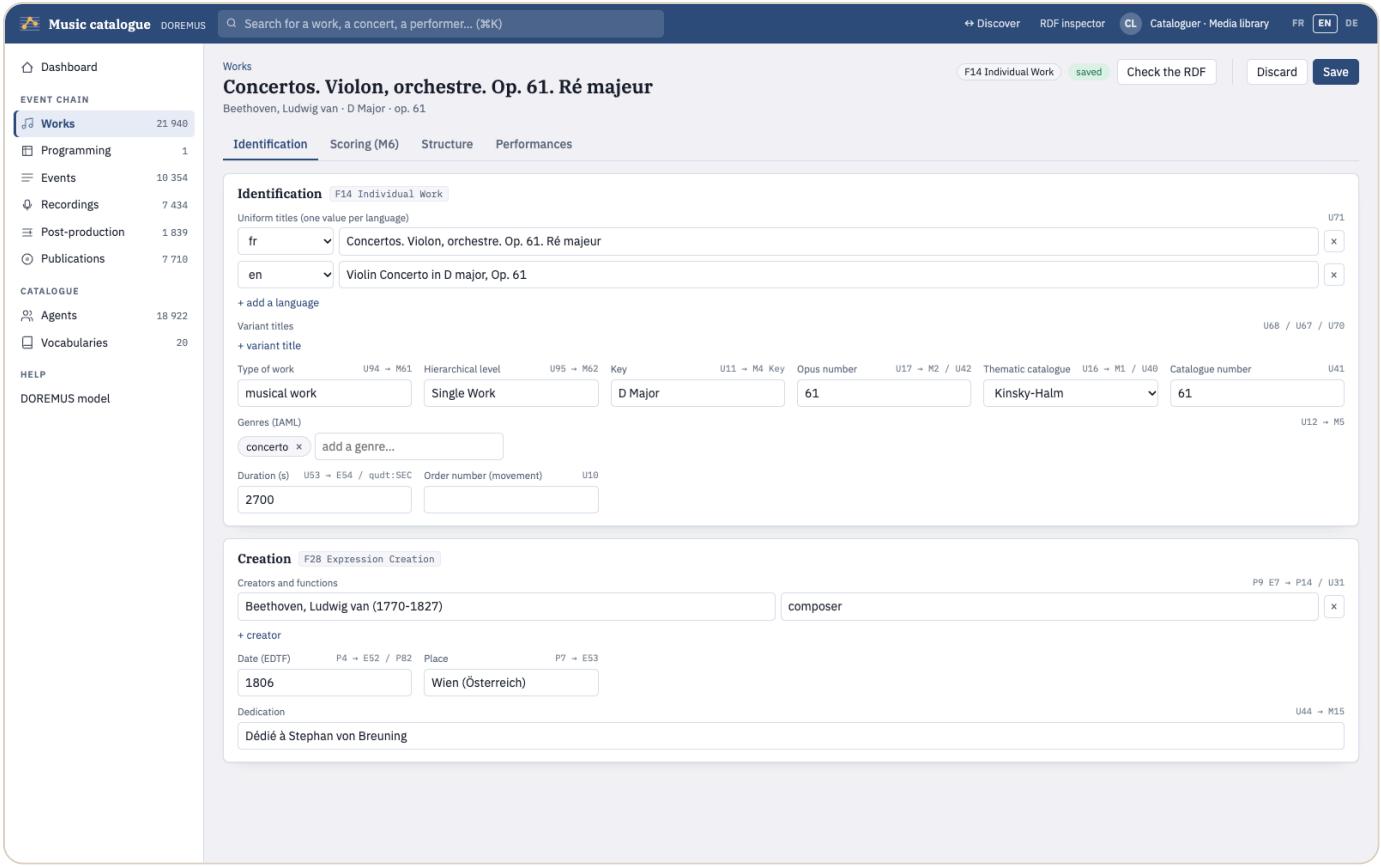
Fields that take a value from a vocabulary (key, genre, instrument, function…) propose the concepts as you type; agents and places are chosen in the same way, and a new one can be created on the spot.

Works

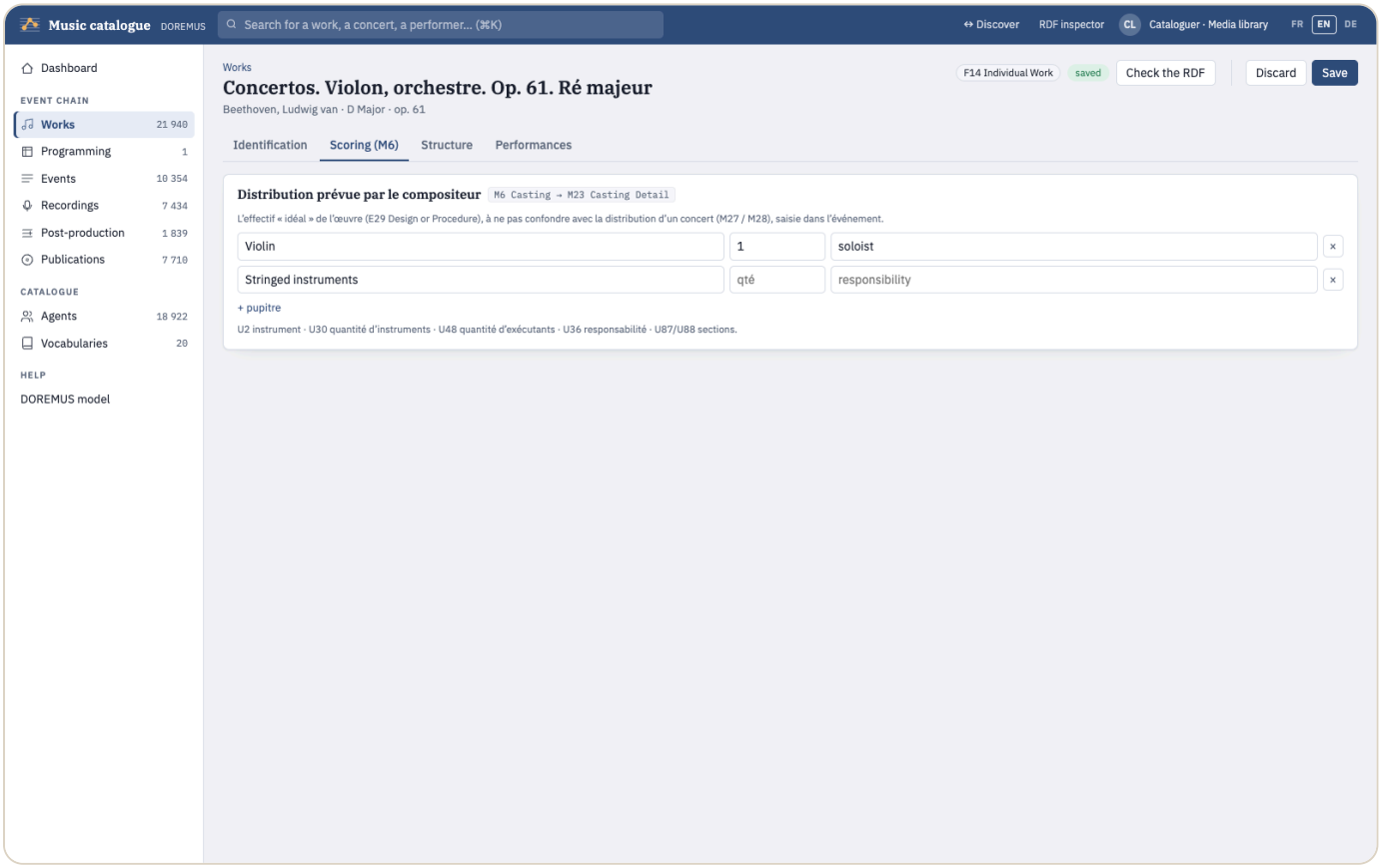
A work record (F14 realised by F22) has four tabs. **Identification**: uniform titles, one per language, variant titles, type and level, key, opus and thematic catalogue number, genres, duration, creation (composer and other creators with their function, date in EDTF, place), dedication. **Scoring (M6)**: the parts planned by the composer, each with its instrument or voice and number. **Structure**: the movements. **Performances**: the events where the work was played.



Works: catalogue records, then the works of data.doremus.org, most performed first. Opening a DOREMUS work copies it into the catalogue to enrich it.



Work record, Identification tab.



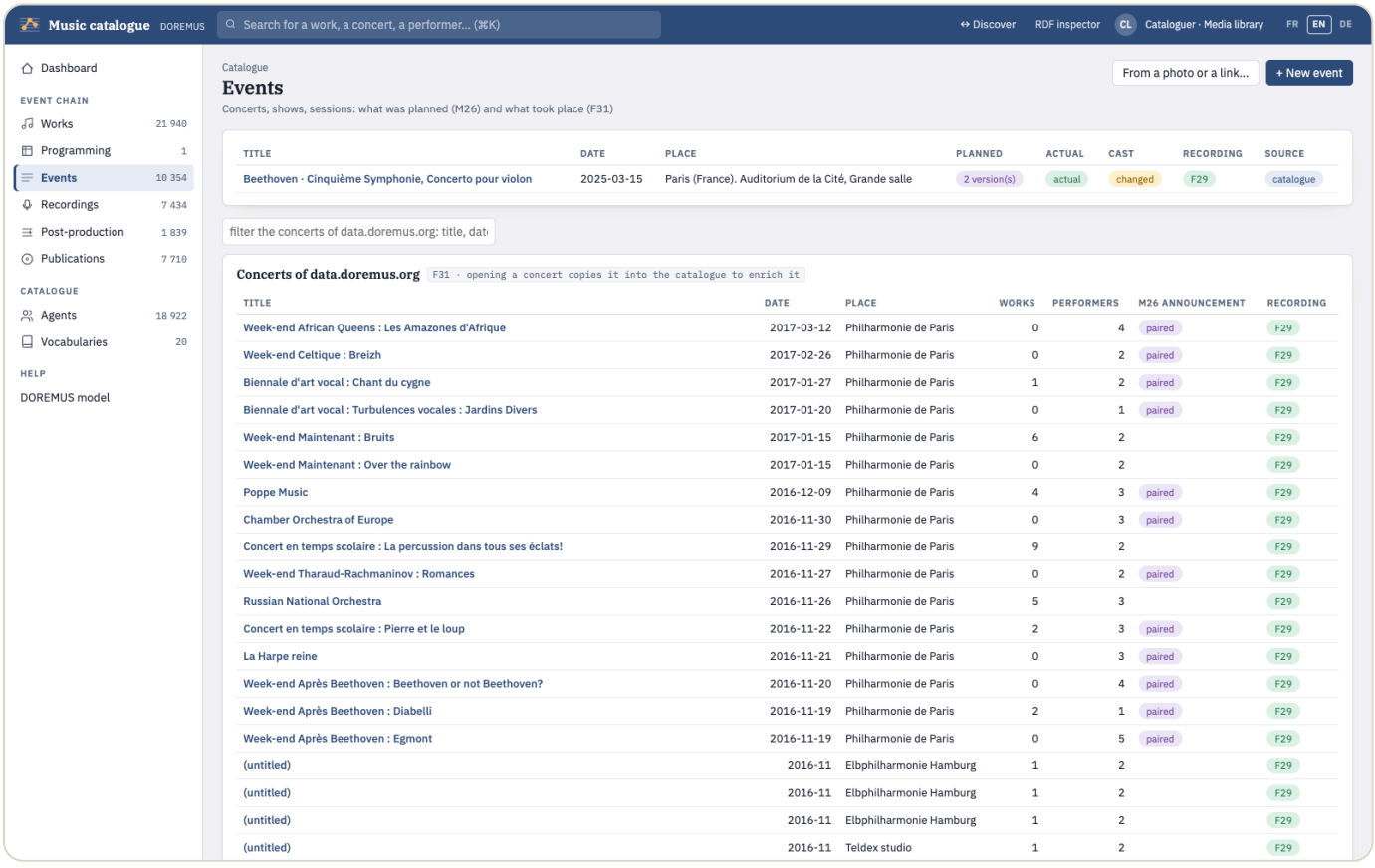
Work record, Scoring tab (M6 Casting → M23 Casting Detail).

Events

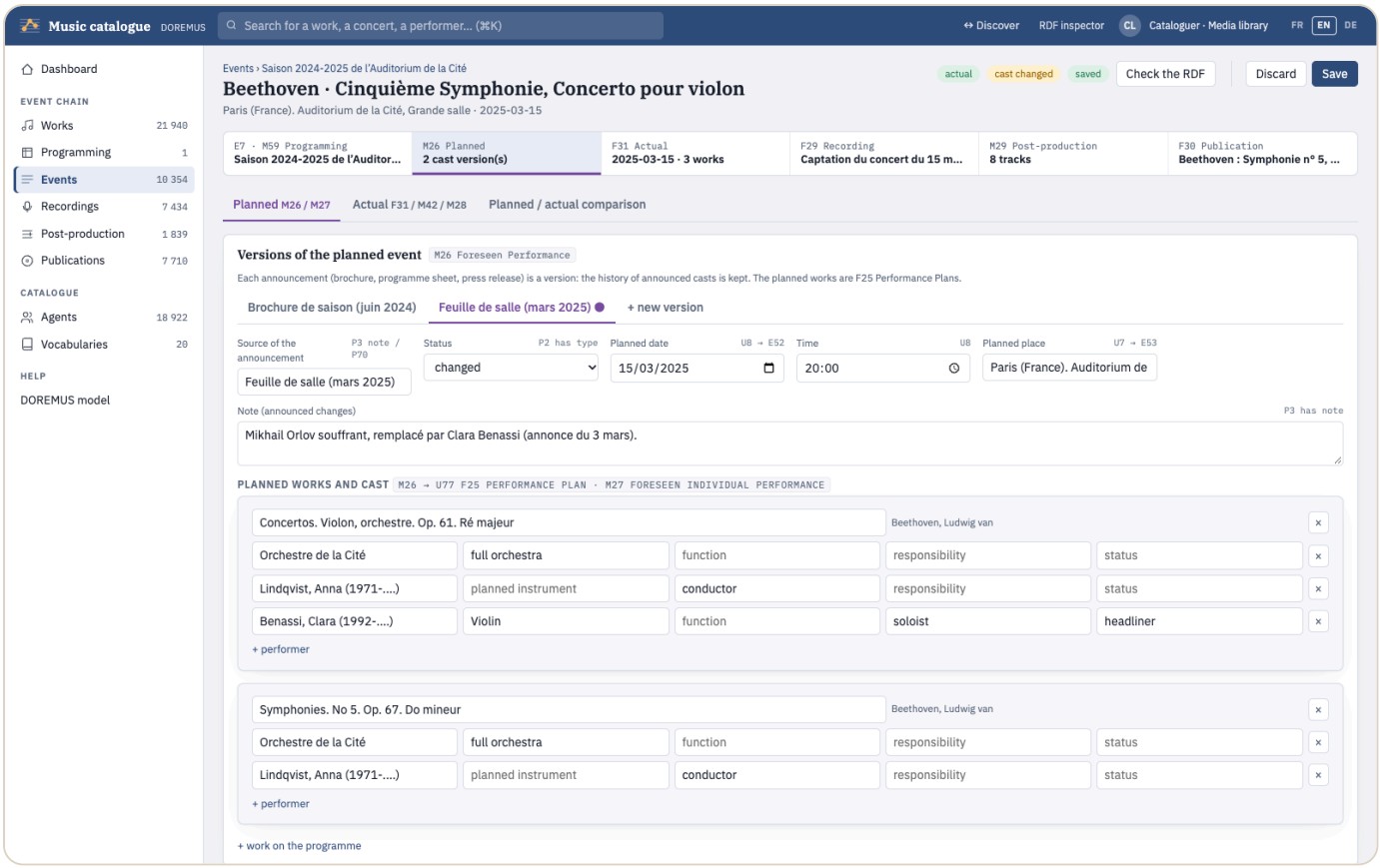
An event separates what was **planned** (M26, one version per announcement: brochure, programme sheet, press release) from what **actually happened** (F31). The chain at the top links the event to its programming, recording, post-production and publication.

- **Planned**: source, status (planned, changed, cancelled, postponed), date, time, place, the works planned and their cast (M27). `+ new version` copies the last announcement to record a change.
- **Actual**: actual date and time, place, conditions, then each work played with its duration, mode and the individual performances (M28: performer, instrument, function, responsibility, status). Encores are works outside the programme.
- **Planned / actual comparison**: replacements (same role, another performer), additions and removals, side by side.

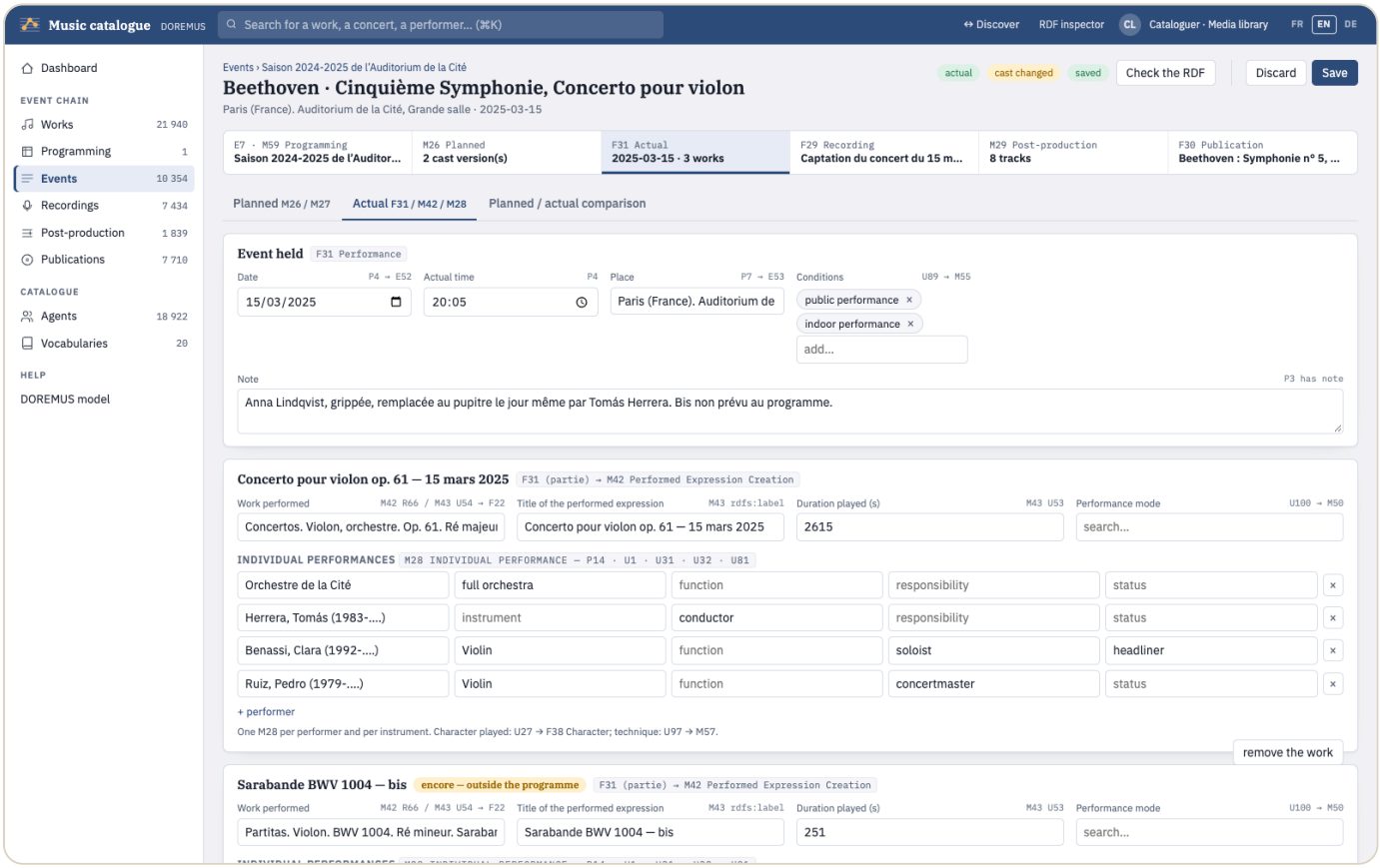
A concert of data.doremus.org opens as a pre-filled event card (planned and actual), ready to be enriched and saved.



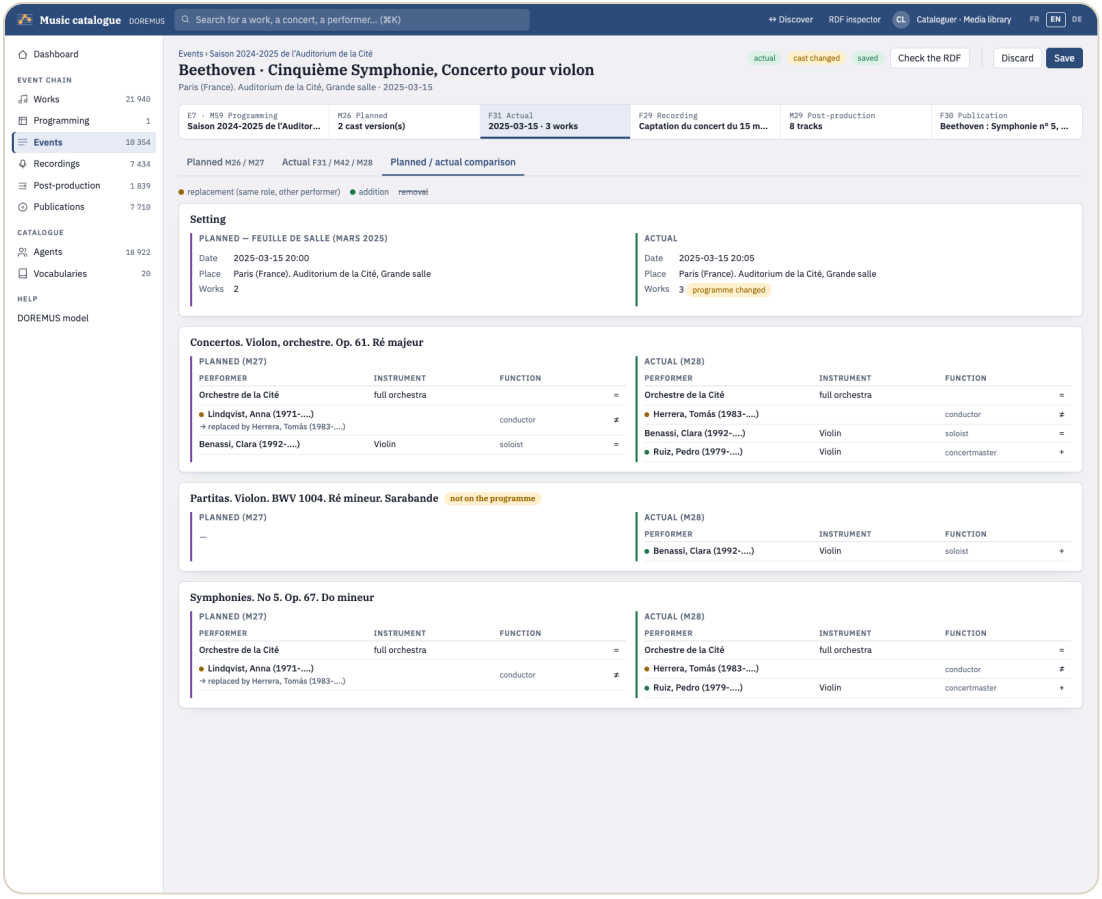
Events: catalogue events, then the concerts and the announcements of data.doremus.org.



Event, Planned tab: versions of the announcement and planned cast.



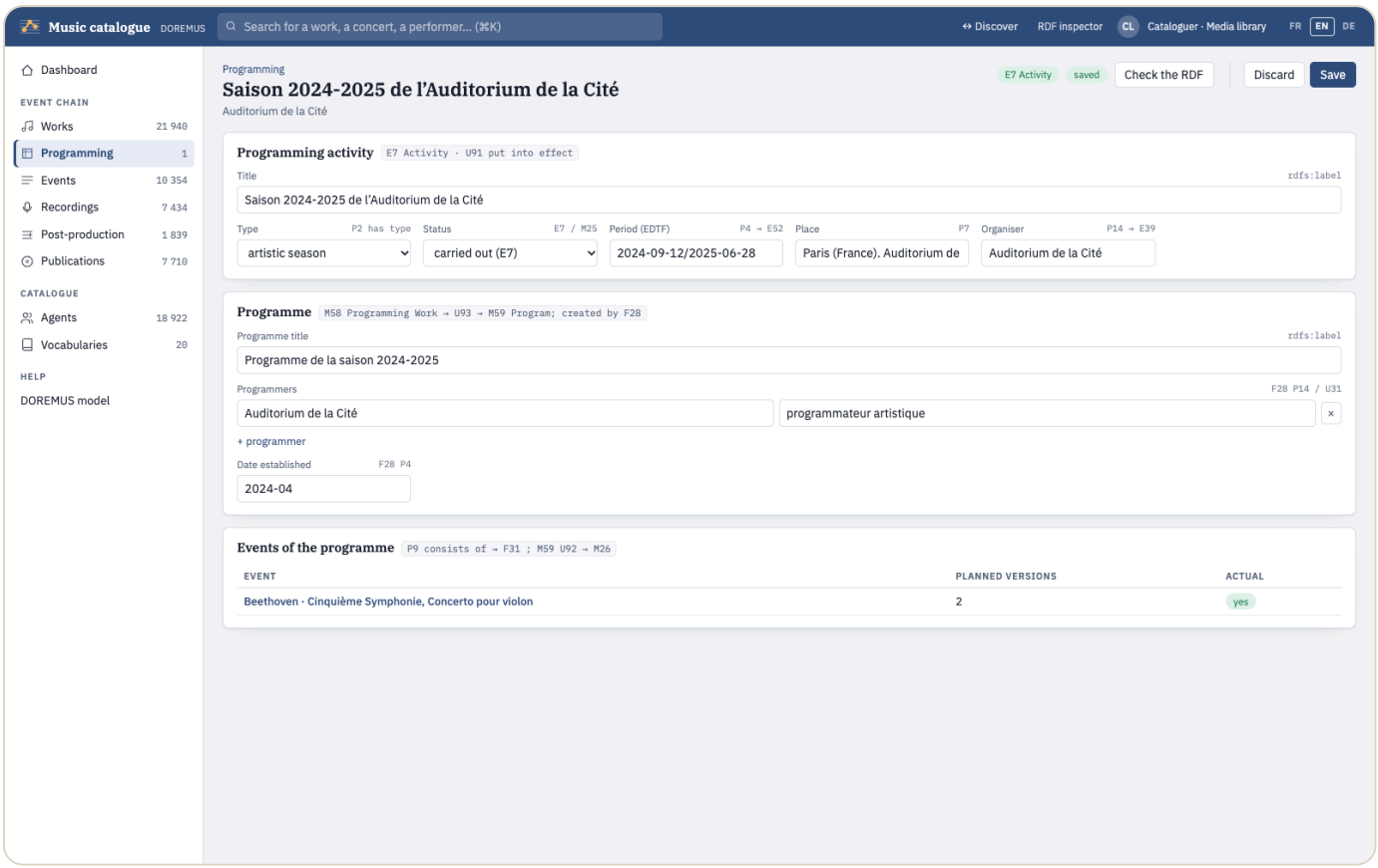
Event, Actual tab: works played and individual performances.



Planned / actual comparison: the conductor announced was replaced (orange), a concertmaster and an encore were added (green).

Programming

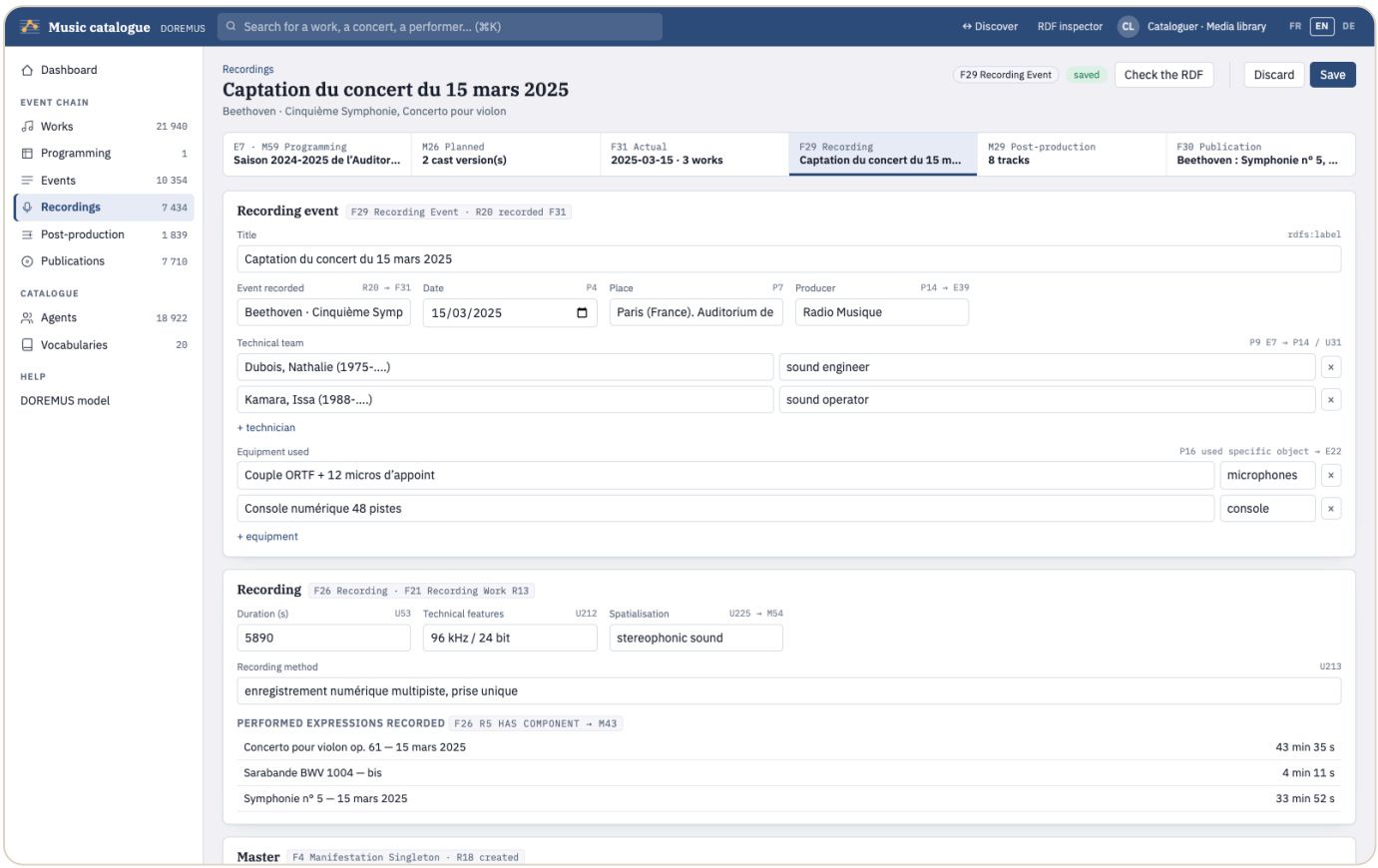
Seasons, festivals, series and tours: a programming activity (E7, or M25 when it is only planned) puts a programme (M59) into effect; the programme lists its events.



Programming record: type, period, organiser, programme and its events.

Recordings

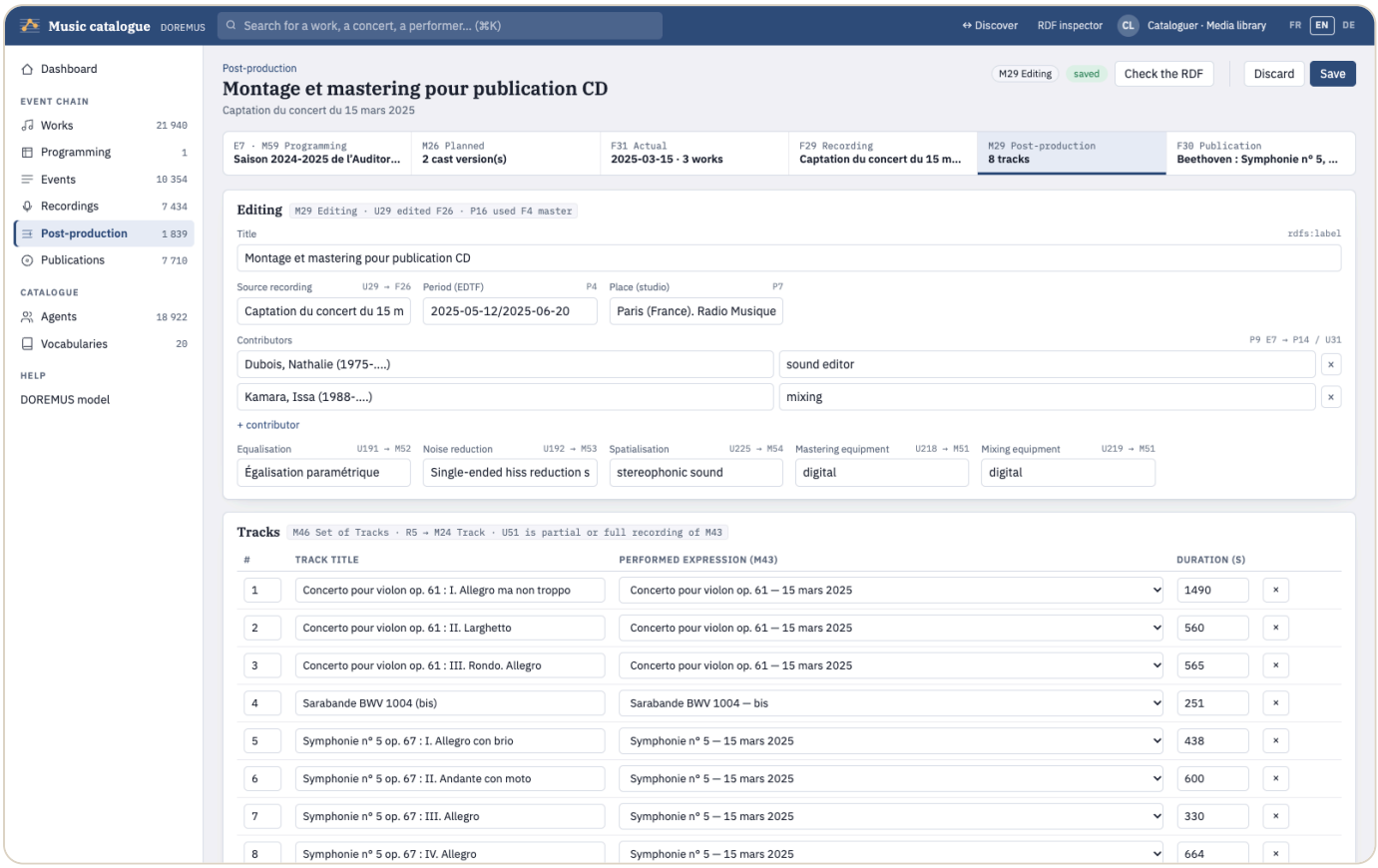
A recording event (F29) records a concert (F31) and creates a recording (F26) and its master (F4): producer, technical team, equipment, duration, technical features, spatialisation, carrier of the master. A recording of data.doremus.org opens as a pre-filled recording card, with the recorded concert, the producer and the place cited by their IRI.



Recording record.

Post-production

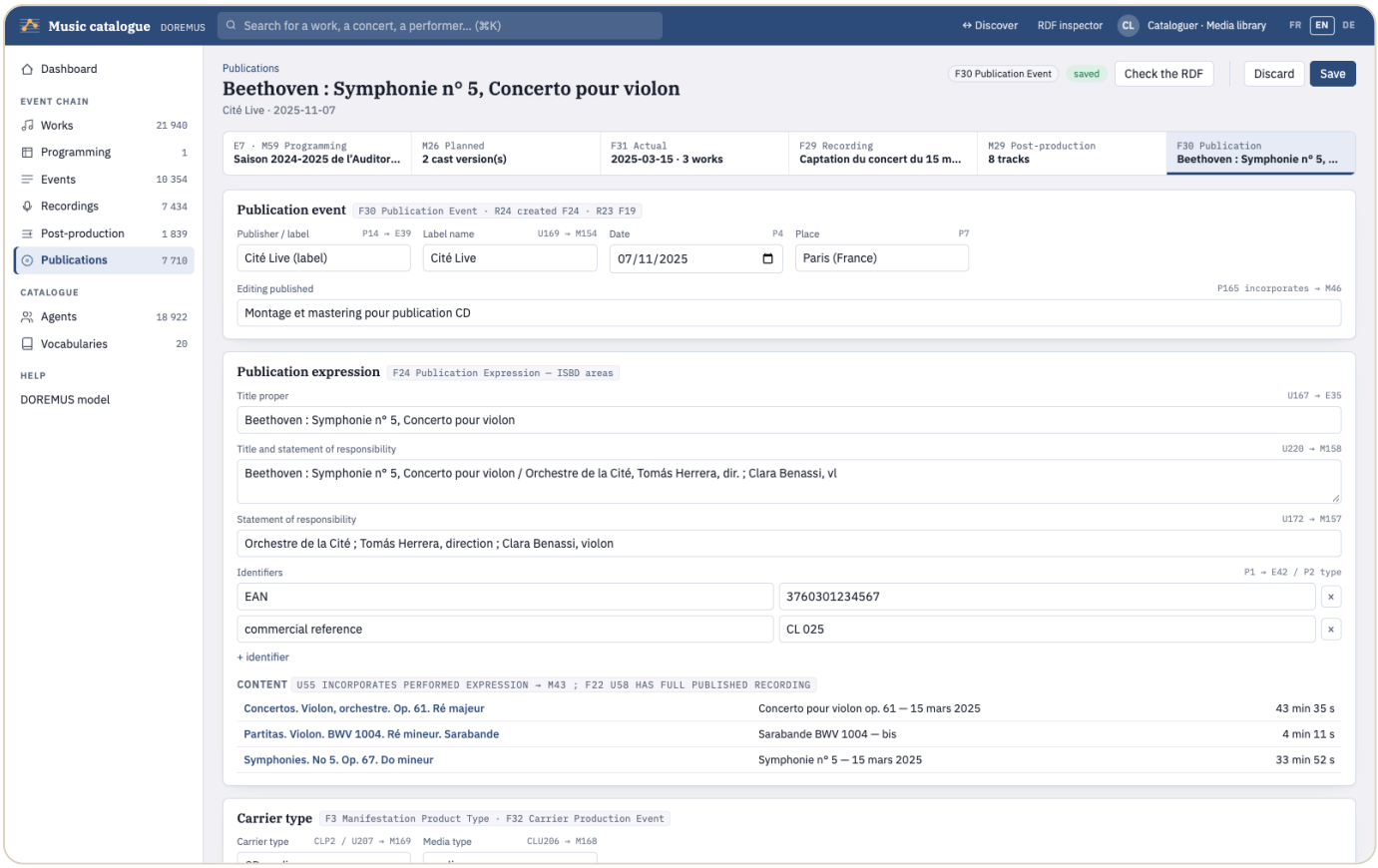
An editing (M29) edits a recording and creates a set of tracks (M46): contributors, equalisation, noise reduction, spatialisation, mastering and mixing equipment, and the tracks with their order, title, duration and the performed work each one records.



Post-production record with its tracks.

Publications

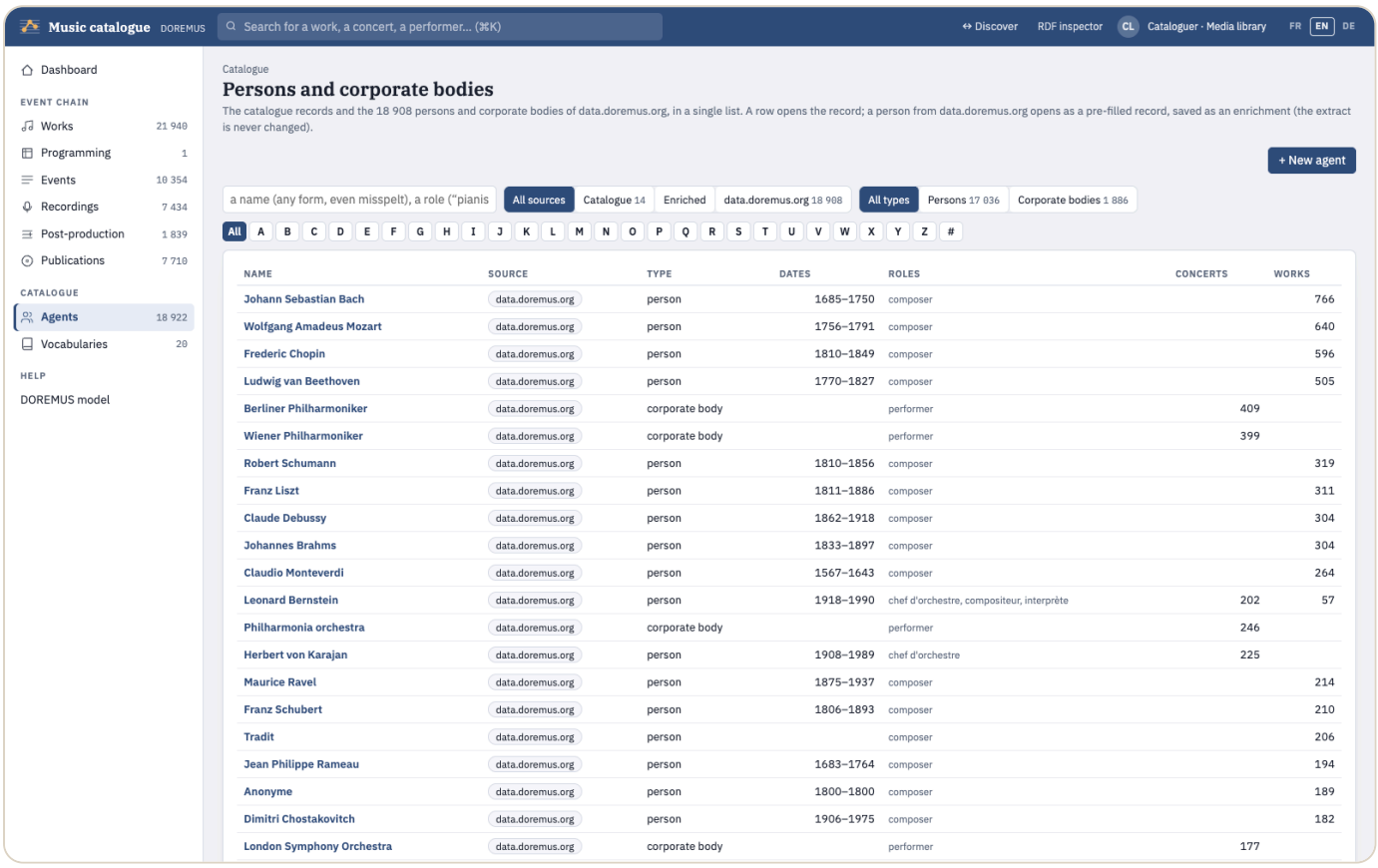
A publication event (F30) creates a publication expression (F24) carried by a product type (F3) and its copies (F5): label, date, title proper and statements of responsibility (ISBD areas), identifiers such as the EAN, carrier, extent, duration, and the copies with their shelfmark, holder and status.



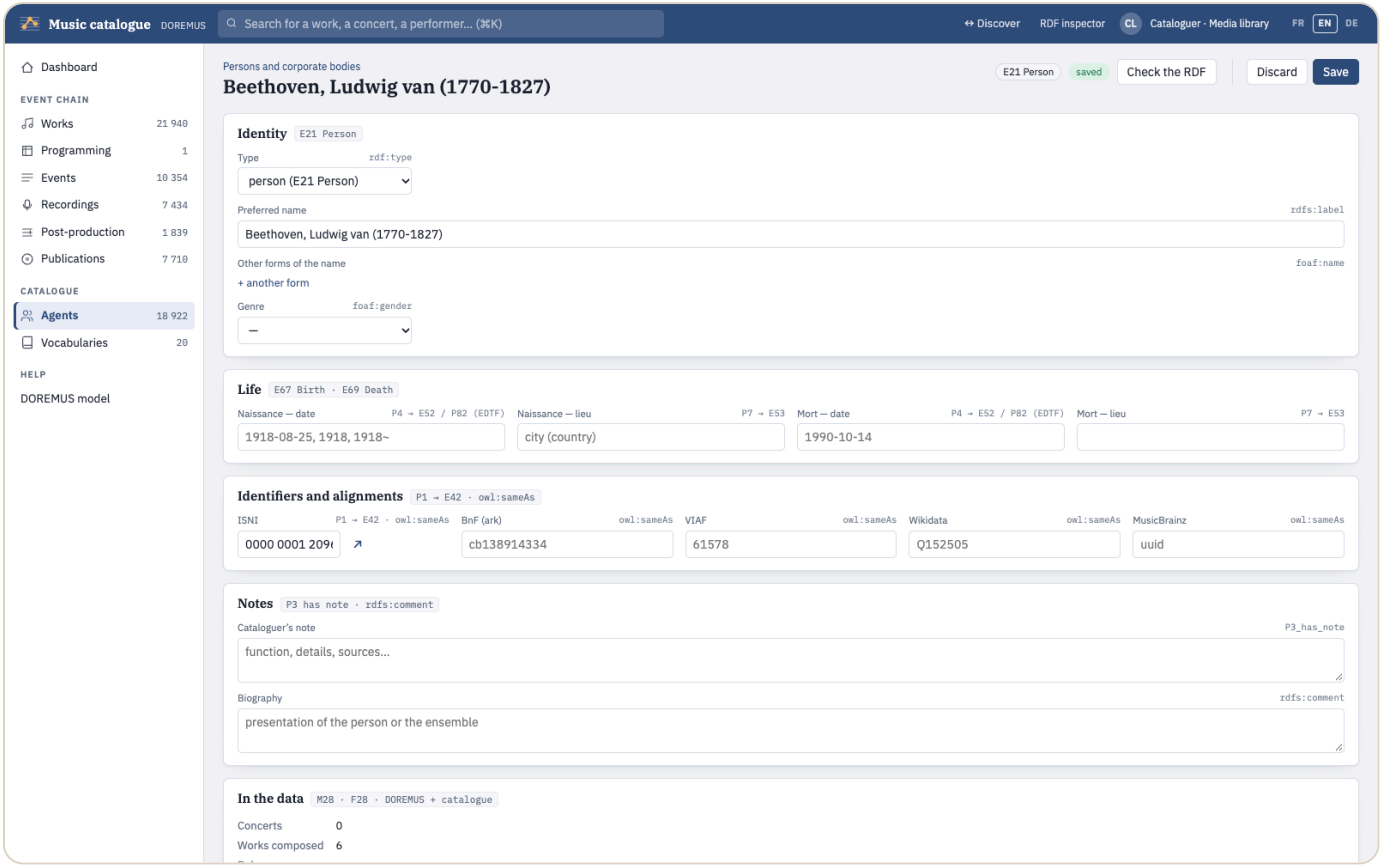
Publication record.

Agents

One list for the catalogue records and the persons and corporate bodies of data.doremus.org, searchable by any form of the name, by role and by A–Z index. A record describes a person (E21), a group (E74) or a corporate body (F11): preferred name in the BnF form, other forms, gender, birth and death (date in EDTF, place), identifiers and alignments (ISNI, BnF, VIAF, Wikidata, MusicBrainz), cataloguer's note and biography. A person of data.doremus.org opens as a pre-filled record and is saved as an enrichment.



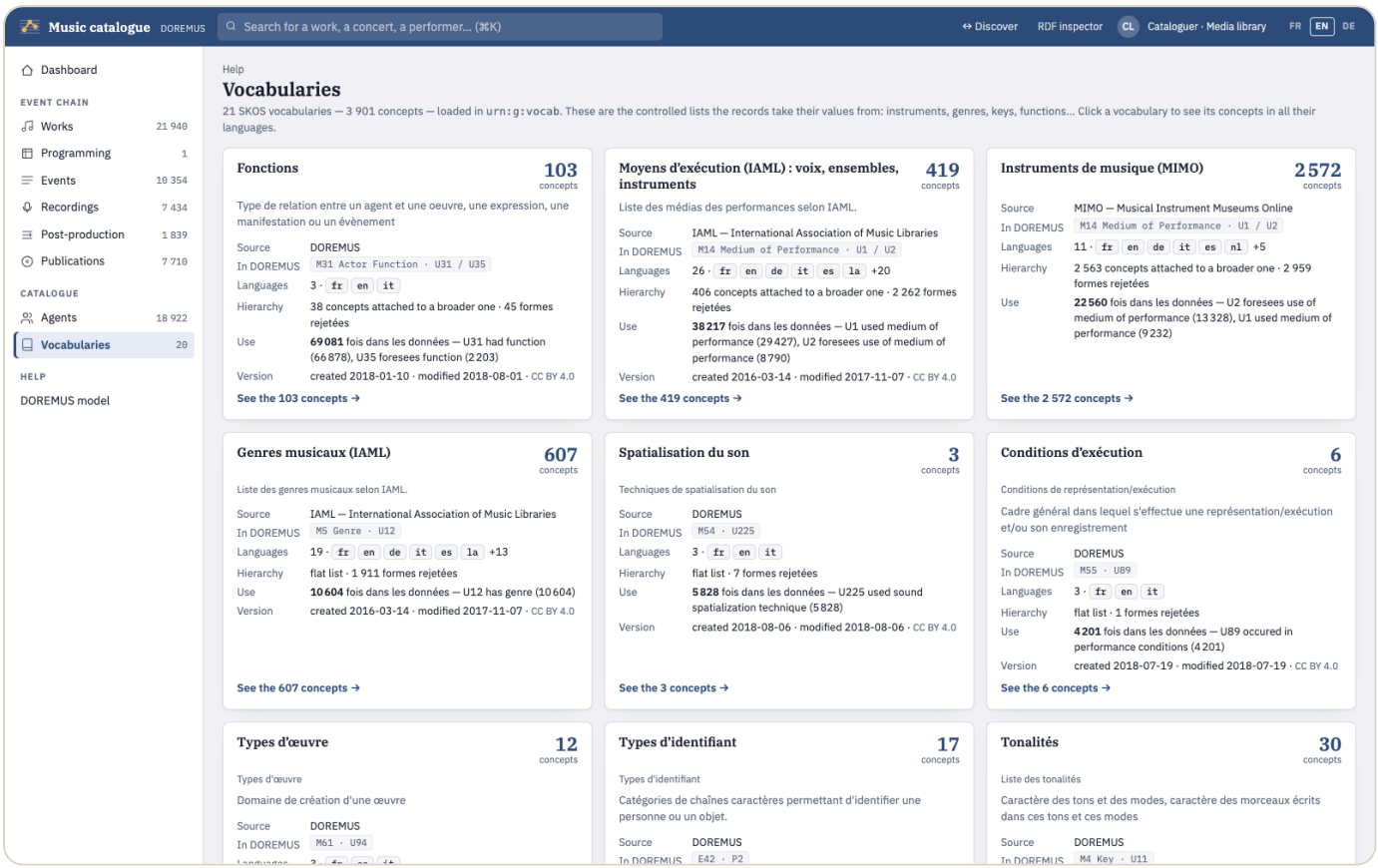
Agents: one list, with sources, types and the A–Z index.



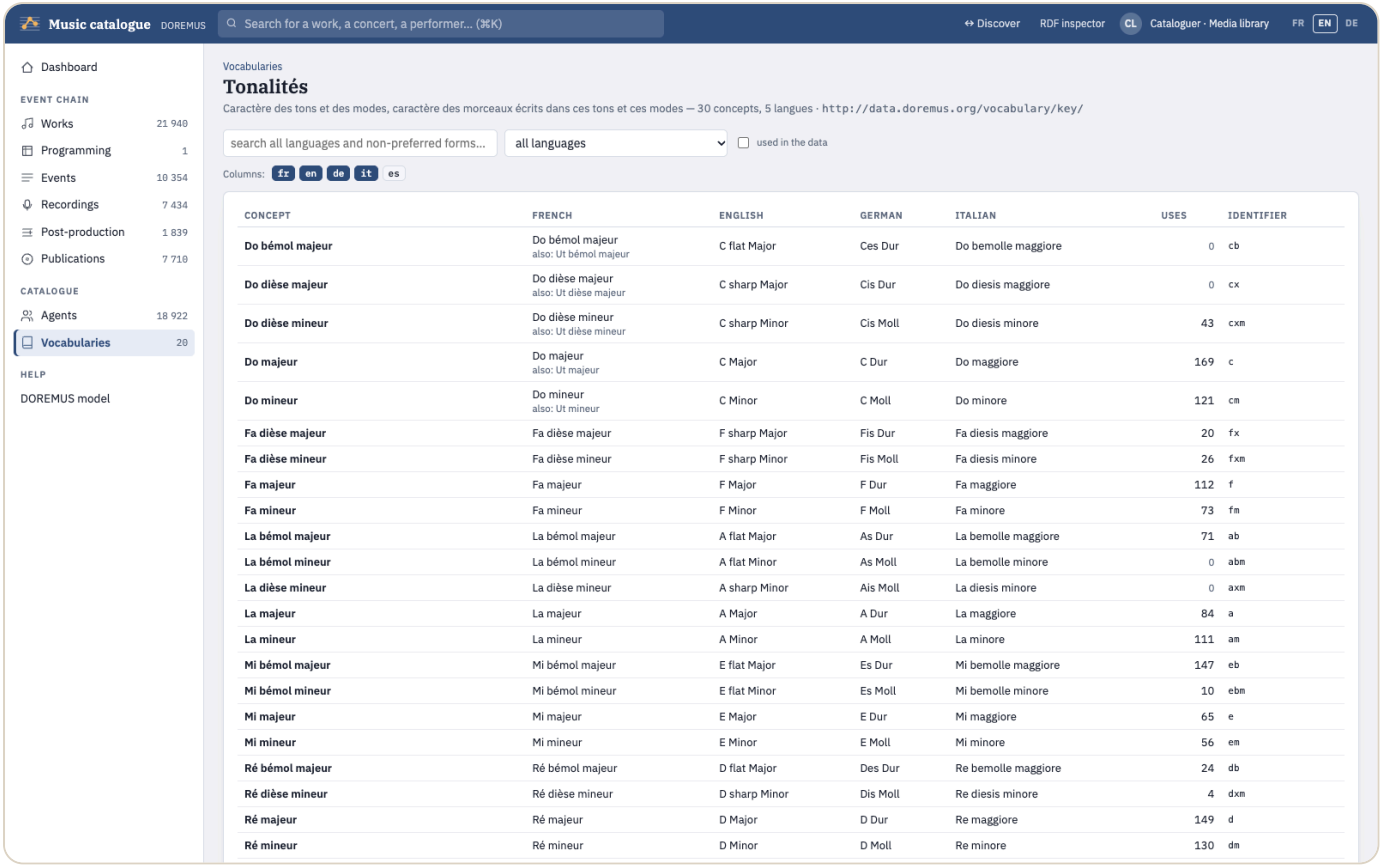
Agent record: identity, life, identifiers and alignments, notes, and what the data says about the person.

Vocabularies

The SKOS vocabularies the records take their values from: instruments (MIMO and IAML), genres, keys, functions… The first page lists them with their source, languages, hierarchy and how much the data uses them; a vocabulary opens on its concepts in every language, in hierarchical order, with a search across all languages and non-preferred forms.



Vocabularies.

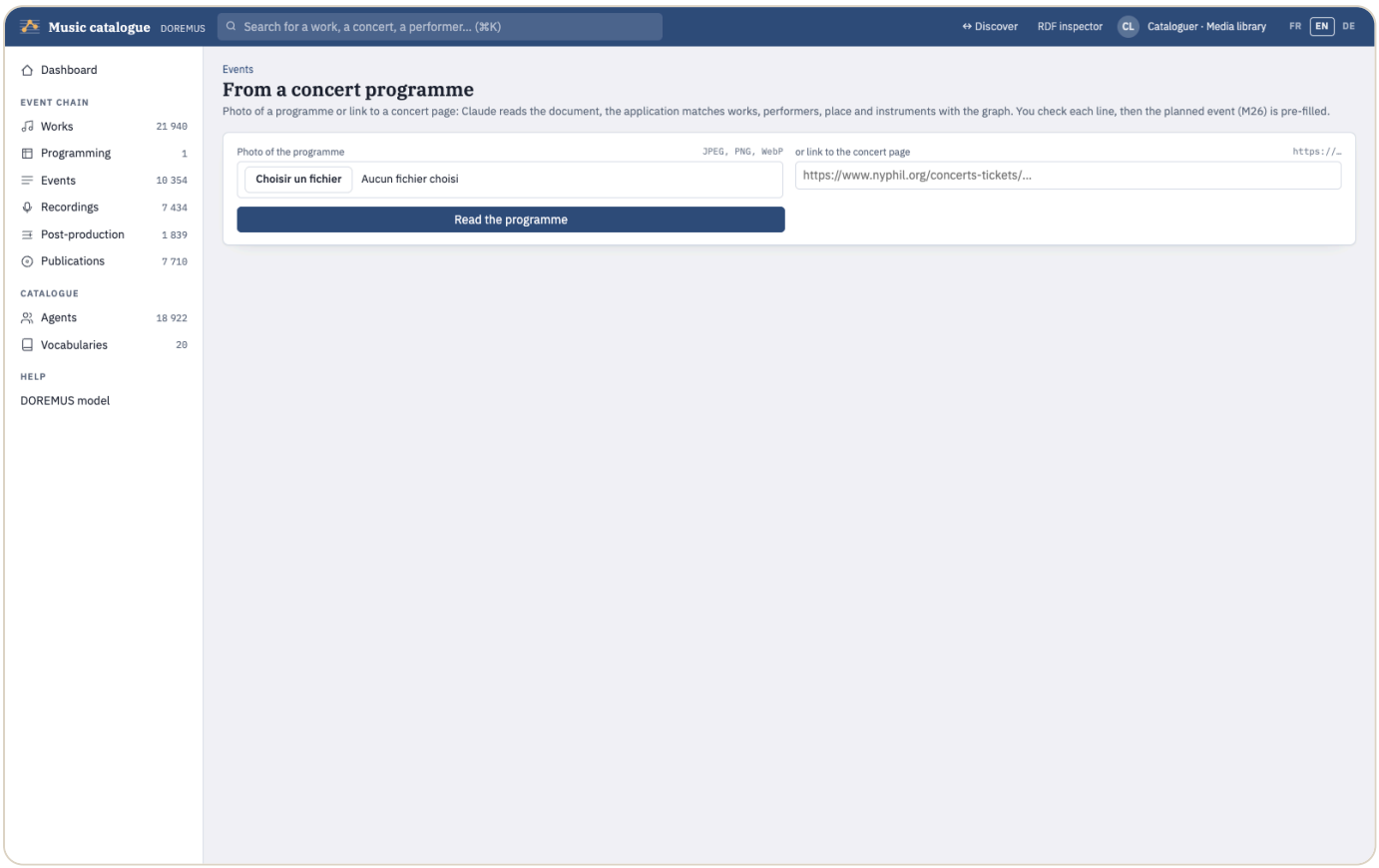


CONCEPT	FRENCH	ENGLISH	GERMAN	ITALIAN	USES	IDENTIFIER
Do bémol majeur	Do bémol majeur also: Ut bémol majeur	C flat Major	Ces Dur	Do bemolle maggiore	0	cb
Do dièse majeur	Do dièse majeur also: Ut dièse majeur	C sharp Major	Cis Dur	Do diesis maggiore	0	cx
Do dièse mineur	Do dièse mineur also: Ut dièse mineur	C sharp Minor	Cis Moll	Do diesis minore	43	cxm
Do majeur	Do majeur also: Ut majeur	C Major	C Dur	Do maggiore	169	c
Do mineur	Do mineur also: Ut mineur	C Minor	C Moll	Do minore	121	cm
Fa dièse majeur	Fa dièse majeur	F sharp Major	Fis Dur	Fa diesis maggiore	20	fx
Fa dièse mineur	Fa dièse mineur	F sharp Minor	Fis Moll	Fa diesis minore	26	fxm
Fa majeur	Fa majeur	F Major	F Dur	Fa maggiore	112	f
Fa mineur	Fa mineur	F Minor	F Moll	Fa minore	73	fm
La bémol majeur	La bémol majeur	A flat Major	As Dur	La bemolle maggiore	71	ab
La bémol mineur	La bémol mineur	A flat Minor	As Moll	La bemolle minore	0	abm
La dièse mineur	La dièse mineur	A sharp Minor	Ais Moll	La diesis minore	0	axm
La majeur	La majeur	A Major	A Dur	La maggiore	84	a
La mineur	La mineur	A Minor	A Moll	La minore	111	am
Mi bémol majeur	Mi bémol majeur	E flat Major	Es Dur	Mi bemolle maggiore	147	eb
Mi bémol mineur	Mi bémol mineur	E flat Minor	Es Moll	Mi bemolle minore	10	ebm
Mi majeur	Mi majeur	E Major	E Dur	Mi maggiore	65	e
Mi mineur	Mi mineur	E Minor	E Moll	Mi minore	56	em
Ré bémol majeur	Ré bémol majeur	D flat Major	Des Dur	Re bemolle maggiore	24	db
Ré dièse mineur	Ré dièse mineur	D sharp Minor	Dis Moll	Re diesis minore	4	dxm
Ré majeur	Ré majeur	D Major	D Dur	Re maggiore	149	d
Ré mineur	Ré mineur	D Minor	D Moll	Re minore	130	dm

The concepts of the « Keys » vocabulary in each language.

From a concert programme

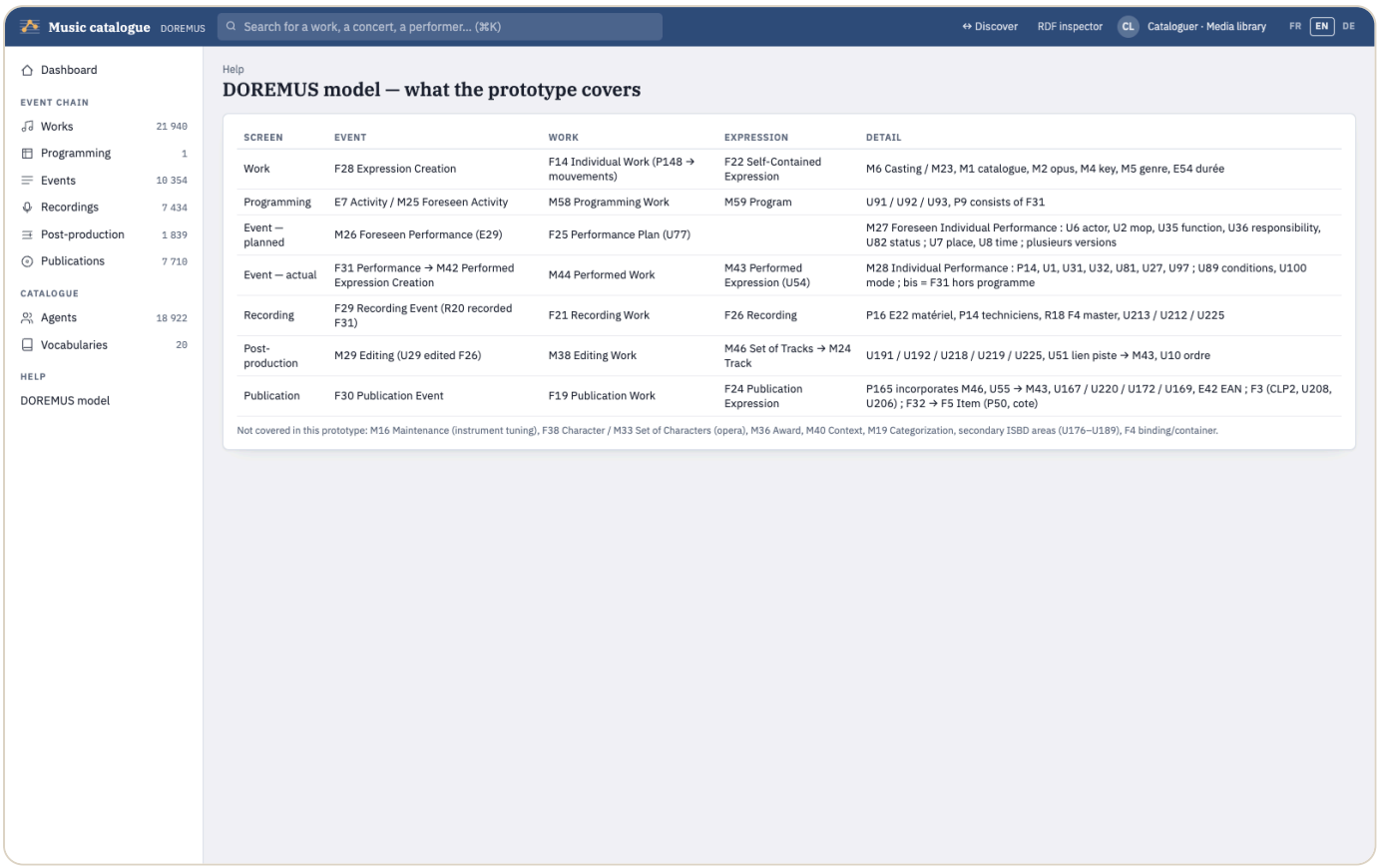
Give a photo of a concert programme (JPEG, PNG, WebP) or a link to a concert page: Claude reads the document, and the application matches the works, performers, place and instruments with the graph. You check each line — choose a candidate or create a new record — then the planned event (M26) is pre-filled.



From a concert programme: photo or link.

DOREMUS model

The Help section of the menu has a page that maps each screen to the classes and properties of the model, and lists what the prototype does not cover.



What each screen covers in the DOREMUS model.

Data and sources

GRAPH	CONTENT
`urn:g:doremus`	Extract of data.doremus.org: the most performed works and their performances, Philharmonie concerts, all announcements (M26), plus a jazz extract and a French chanson extract, with their concerts and recordings. Changed only by documented loading steps (genre harmonisation, instrument values, ISNI URIs).
`urn:g:data`	Records created or enriched in the catalogue.
`urn:g:vocab`, `urn:g:vocab-local`	SKOS vocabularies, and local concepts added from the catalogue.
`urn:g:align`	ISNI alignments of persons (ISNI SRU API: name, then dates or years of musical activity).
`urn:g:wikidata`	Wikidata alignments: persons with a BnF record by their BnF identifier, the others by name with the same rules as for ISNI. Adds Wikidata, ISNI, VIAF, MusicBrainz and BnF links, French and English Wikipedia pages and missing life years. Disagreements are not written but listed for review.
`urn:g:prov`	Provenance (PROV) of every save and every loading step.

Administration

On the server, from the application folder (commands of the `Makefile`):

```
make up              # start Fuseki
make dev             # start the application (port 8000)
make load-doremus    # reload the data.doremus.org extracts and their loading steps
make import-jazz     # fetch and add the jazz extract
make import-chanson  # fetch and add the French chanson extract
make isni            # align persons with ISNI
make wikidata        # align persons with Wikidata
make test            # run the tests
```

The questions and the generated texts use the Anthropic API: set `ANTHROPIC_API_KEY` in the `.env` file. Without it, the rest of the application works and text generation falls back on the template.

Model codes

The codes shown beside headings (when « model codes » is ticked) and in the catalogue are those of FRBRoo (F, R), CIDOC-CRM (E, P) and DOREMUS (M, U).

CODE	MEANING
`F14` / `F22`	Individual work / its self-contained expression (titles, key, genre, scoring)
`F28`	Expression creation: composer, date and place of composition
`M6` → `M23`	Casting planned by the composer → casting detail (one part)
`M26` / `M27`	Foreseen performance (an announced concert) / foreseen individual performance
`F31` / `M28`	Performance (a concert, or a work in it) / individual performance (a performer, an instrument, a role)
`M42` → `M43`	Performed expression creation → performed expression (the work as played)
`F29` → `F26`, `F4`	Recording event → recording, master
`M29` → `M46`, `M24`	Editing → set of tracks, track
`F30` → `F24`, `F3`, `F5`	Publication event → publication expression, product type, item
`E7` / `M25`, `M59`	Programming activity (actual / planned), programme
`E21` / `E74` / `F11`	Person / group / corporate body
`E52` / `E53`	Time-span / place

Technical architecture

The application is two programs: a triple store that holds every piece of data, and one Python process that serves the API and the web pages. There is no other database: the records of the catalogue, their editing state, the vocabularies, the ontologies and the provenance are all RDF in the triple store.

Discover music
`explore.html` · read-only services, questions, RDF view

Music catalogue
`app.html` · records, RDF inspector, validation report

User guide
`guide/index.html` · this document

↕ HTTP and JSON (`/api/…`), streamed answers for questions

Application server · FastAPI (Python 3.12)
Read services and discovery index, conversion of records to RDF, SHACL validation, provenance, calls to Claude

Anthropic API
Claude: questions, text generation, reading of concert programmes

↕ SPARQL 1.1 Query and Update

Apache Jena Fuseki 5.6.0
TDB2 store, one named graph per source, default graph = union of the graphs, Lucene full-text index

Loading only
data.doremus.org SPARQL endpoint, ISNI SRU API, Wikidata

The external services in dashed boxes are not needed to browse or catalogue: the data.doremus.org extract and the alignments are loaded once into the store; only questions, generated texts and programme reading call the Anthropic API.

Technology stack

LAYER	TECHNOLOGY	ROLE
Triple store	Apache Jena Fuseki 5.6.0 (Java 21), TDB2	Stores all the RDF in named graphs; SPARQL 1.1 Query, Update and Graph Store Protocol. Built from `fuseki/Dockerfile` and run with Docker Compose, reachable from the machine only.
Full-text search	Jena text + Apache Lucene	Index of `rdfs:label`, `skos:prefLabel` / `altLabel`, uniform titles (U71), titles (P102) and subtitles (U67), insensitive to accents and case; queried in SPARQL with `text:query`.
Application server	Python 3.12, FastAPI 0.156, Uvicorn 0.54	One process for the REST API and the static pages.
RDF and validation	rdflib 7.6, pySHACL 0.40	Builds the RDF of each record and validates it against the SHACL shapes before it is saved.
HTTP client	httpx 0.28	Calls to Fuseki, to data.doremus.org and to the alignment sources.
Language model	Anthropic Python SDK 1.16, Claude Opus 5.5	Questions in plain language (tools: full-text search, read-only SPARQL), generated texts, reading of concert programmes. Optional: set `ANTHROPIC_API_KEY`.
Web pages	HTML, CSS and plain JavaScript	No framework and no build step; French, English and German interface; light and dark themes.
Tooling	uv, Docker Compose, Make, pytest	Python environment, the store's container, the commands of the Administration chapter, the automated tests.

Ontologies

The ontologies are loaded in `urn:g:ontology` (folder `ontology/`); the catalogue writes RDF that respects their domains and ranges.

ONTOLOGY	PREFIX	FILE AND VERSION	USED FOR
CIDOC-CRM, Erlangen OWL version (ECRM)	`ecrm:`	`crm.rdf`, CIDOC-CRM 6.2.1	Events, actors (E21, E74), time-spans (E52), places (E53), titles, identifiers
FRBRoo, Erlangen OWL version (EFRBRoo)	`efrbroo:`	`frbroo.owl`, EFRBRoo 121016 / FRBRoo 1.0.2	Works and expressions (F14, F22, F28), performances (F31), recordings (F29, F26), publications (F30, F24, F5)
DOREMUS ontology	`mus:`	`doremus.ttl`, modified March 2024	The music extension: casting (M6, M23), foreseen performances (M26, M27), individual performances (M28), editing and tracks (M29, M46, M24), programming (M59), keys, genres, opus and catalogue statements
SKOS	`skos:`	W3C	The controlled vocabularies and their hierarchies (`skos:broader`)
PROV-O	`prov:`	W3C	Provenance of every save, deletion and loading step
OWL-Time	`time:`	W3C	Dates of data.doremus.org (`time:hasBeginning`, `time:inXSDDate`)
OWL	`owl:`	W3C	`owl:sameAs` between an enriched record and the data.doremus.org original
Schema.org, GeoNames, WGS84	`schema:`, `gn:`, `geo:`	as used by data.doremus.org	Birth and death dates of persons; cities and coordinates of places

Vocabularies and shapes

- **Vocabularies** (`vocabularies/` , graph `urn:g:vocab`): the 23 SKOS vocabularies of the DOREMUS project, among them instruments MIMO and IAML, genres IAML, keys, modes, functions, responsibilities, performance conditions, carriers and identifiers. Concepts added from the catalogue go to `urn:g:vocab-local` ; the DOREMUS vocabularies are never modified. At load, `skos:topConceptOf` is completed with `skos:inScheme` .
- **SHACL shapes** (`shapes/`): `work-card.ttl` (works, expressions, creation, casting), `event-card.ttl` (announced and actual concerts), `chain-cards.ttl` (programming, recording, editing, publication, agents). A violation blocks the save; a warning is saved and shown.

How the data moves

- **Reading.** Every query is a SPARQL file in `backend/sparql/` ; values are inserted as checked RDF terms, never as raw text. The discovery services work on an index built in memory from these queries when the server starts, and rebuilt after each save.
- **Saving a record.** The form is sent as JSON, turned into RDF (rdflib), validated (pySHACL), then written in one SPARQL Update: the triples to `urn:g:data` , the form state to `urn:g:cards` , a PROV activity to `urn:g:prov` .
- **Questions.** Claude receives the question and five tools — full-text search, work and person summaries, the RDF of a resource, and read-only SPARQL (SELECT or ASK, row limit, time limit). It writes the queries itself from a description of the real shapes of the data, and its answer cites the IRIs it found.
- **Loading.** Scripts in `scripts/` fetch the data.doremus.org extracts (paged CONSTRUCT queries, at most 50 IRIs per request), align persons with ISNI and Wikidata, and record each step in PROV.

Code layout

PATH	CONTENT
`backend/main.py`	Application entry point: API routes and static pages
`backend/store.py`, `backend/sparql/`	Access to Fuseki and the SPARQL queries
`backend/corpus.py`, `services.py`, `people.py`, `resource.py`	Discovery index, discovery services, people search, generic RDF view
`backend/cards.py`, `records.py`, `validation.py`, `importer.py`	Catalogue records: RDF conversion, saving, SHACL validation, pre-filled records from data.doremus.org
`backend/vocab.py`, `vocabularies.py`	Controlled vocabularies
`backend/chat.py`, `textgen.py`, `programme.py`	Features that use Claude
`ui/web/`	The web pages, the interface translations and this guide
`ontology/`, `vocabularies/`, `shapes/`	Ontologies, SKOS vocabularies, SHACL shapes
`scripts/`, `fuseki/`, `tests/`	Loading and alignment scripts, the store's container and configuration, automated tests

Screenshots taken from the application in English, October 2026.