

Chat Solution

Chat with prompts management, from PostgreSQL to clients and console.

Purpose

- Demonstrate chat and reusable prompt taxonomy CRUD.
- Keep the public object separate from JPA persistence.
- Share one HTTP contract across Java, TypeScript, and Dart.
- Include a practical React console and bulk data workflows.

Runtime

- Spring Boot APIs: `/api/v1/chat`, `/api/v1/task`, `/api/v1/tag`, `/api/v1/nomenclature`.
- PostgreSQL persistence through Spring Data JPA.
- Java, TypeScript, and Dart clients.
- React, Vite, and MUI management console.
- MkDocs documentation and Marp presentation in the same repo.

Task Contract

TODO

Field	Rule
id	Server-generated Long
code	Required, unique, immutable string
label	Required string
description	Nullable string
status	TODO / IN_PROGRESS / DONE / CANCELLED
priority	LOW / NORMAL / HIGH / URGENT
Due / assignee	Nullable YYYY-MM-DD date / string
Taxonomy	tagIds: [], nullable nomenclatureId
Audit	Server-owned timestamps and actors

Defaults: status TODO , priority NORMAL , no taxonomy assignment.

Task

TODO

```
{
  "id": 42,
  "code": "RELEASE-001",
  "label": "Prepare release",
  "description": "Review the release checklist",
  "status": "TODO",
  "priority": "HIGH",
  "dueDate": "2026-11-01",
  "assignedTo": "alice",
  "tagIds": [],
  "nomenclatureId": null,
  "createdDate": "2026-08-30T17:00:00Z",
  "lastModifiedDate": "2026-08-30T17:05:00Z",
  "createdBy": "admin",
  "lastModifiedBy": "alice"
}
```

API

TODO

Method	Path
CRUD	<code>/api/v1/task[/]{id}]</code>
CRUD	<code>/api/v1/tag[/]{id}]</code>
CRUD	<code>/api/v1/nomenclature[/]{id}]</code>
POST	<code>/api/v1/task/seed</code>
POST	<code>/api/v1/task/import.{json,csv}</code>
GET	<code>/api/v1/task/export.{json,csv}</code>
PDF	<code>/api/v1/task/{id}/print.pdf</code> , <code>/print.pdf</code> , <code>/print-selected.pdf</code>

Seed and import upsert by `code` and ignore imported IDs and dates.

Create includes `code` ; update excludes immutable `code` .

Reusable Taxonomy Domains

- **Tag**: flat labels shared by tasks, such as Release or Documentation.
- **Nomenclature**: classification tree, such as Engineering → Release.
- Both: `id`, `code`, `label`, `description`, creation/modification dates and actors.
- Nomenclatures add nullable `parentId`; `null` means a root.
- References must exist; self-parenting and hierarchy cycles are rejected.
- Referenced tags, task classifications, and parents with children cannot be deleted.

Model Boundary

- `Task`, `Tag`, `Nomenclature` : public business objects.
- Internal mutable JPA entities isolate persistence behavior.
- The service maps between public and persistence models.
- Namespace `fr.tiogars.chat` and artifact identities stay unchanged.

Demo Database

Apply `java/api/src/main/resources/db/manual/task-domains.sql` before API startup.

`ddl-auto: update` is not a data migration and does not remove legacy tables.

```
docker compose -f docker/compose.yml -f docker/compose.local.yml down --volumes
docker compose -f docker/compose.yml -f docker/compose.local.yml up -d postgres
```

Resetting the demo volume removes all stored data.

Reapply the task-domain migration, then run `make docker-up`.

Deploy Quickly

```
make docker-up
```

- API: <http://localhost:8989>
- Swagger: <http://localhost:8989/swagger-ui.html>
- Console: <http://localhost:5173>
- Docs: <http://localhost:8000>

Take Away

- Presentation deck: <http://localhost:8082>
- PDF: [chat.pdf](#)
- PPTX: [chat.pptx](#)
- Presenter mode: press **p** on the deck.