



ICT Training Center

Il tuo partner per la Formazione e la Trasformazione digitale della tua azienda



Note



SPRING AI

GENERATIVE ARTIFICIAL INTELLIGENCE CON JAVA

Simone Scannapieco

Corso avanzato per Venis S.p.A, Venezia, Italia

Novembre 2025

Note



RETRIEVAL AUGMENTED GENERATION

PARTE 1

Note

 Storico della conversazione valida per singola sessione

Note

- Configurazione JDBC H2 *chat memory* per ChatClient Ollama
 - 1 Modifica al pom.xml per dipendenze Spring AI JDBC, H2 e devtools
 - 2 Modifica ad application.yml
 - 3 Creazione schema H2
 - 4 Modifica ai bean ChatClient per Ollama
 - 5 Test delle funzionalità con Postman/Insomnia

Note

AMBIENTE DI SVILUPPO CREAZIONE ACCOUNT HUGGINGFACE

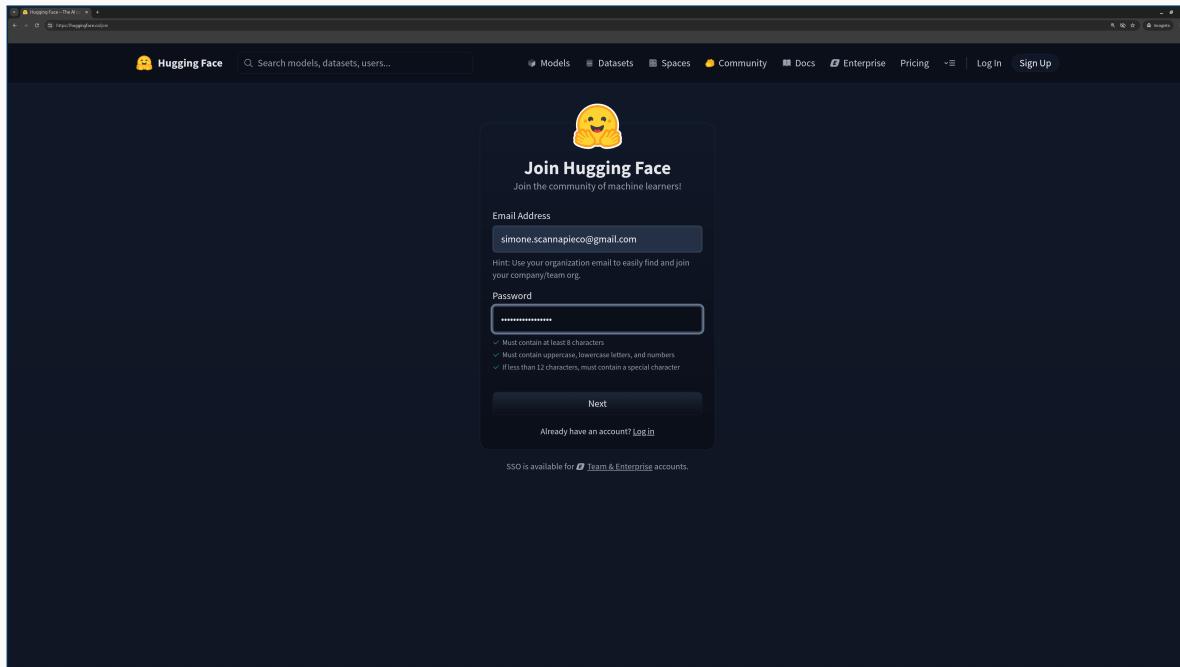
1 Accedere al portale <https://huggingface.co/>

The screenshot shows the Hugging Face website homepage. At the top, there's a navigation bar with links for Models, Datasets, Spaces, Community, Docs, Enterprise, Pricing, Log In, and Sign Up. A search bar is also at the top. The main content area features a large yellow smiley face icon and the text "The AI community building the future." Below this, there's a section for "Explore AI Apps" and "Browse 1M+ models". The right side of the page displays a list of AI models, each with a thumbnail, name, description, and update information. The models listed include: mbert-large/Llama-2-70B, stabilityai/stable-diffusion-v2-base-0.9, openchat/openchat, llyysvval/ControlNet-v2.1, catperso/zeroshot-v2.0, neto-1/lama-2-13B, tisseur/falcon-40b-instruct, WizardM/WizardCoder-15B-v1.0, CompVis/stable-diffusion-v1-4, stabilityai/stable-diffusion-2-1, Salesforce/gpt-7b-8k-inst, and THUDU/Update.

Note

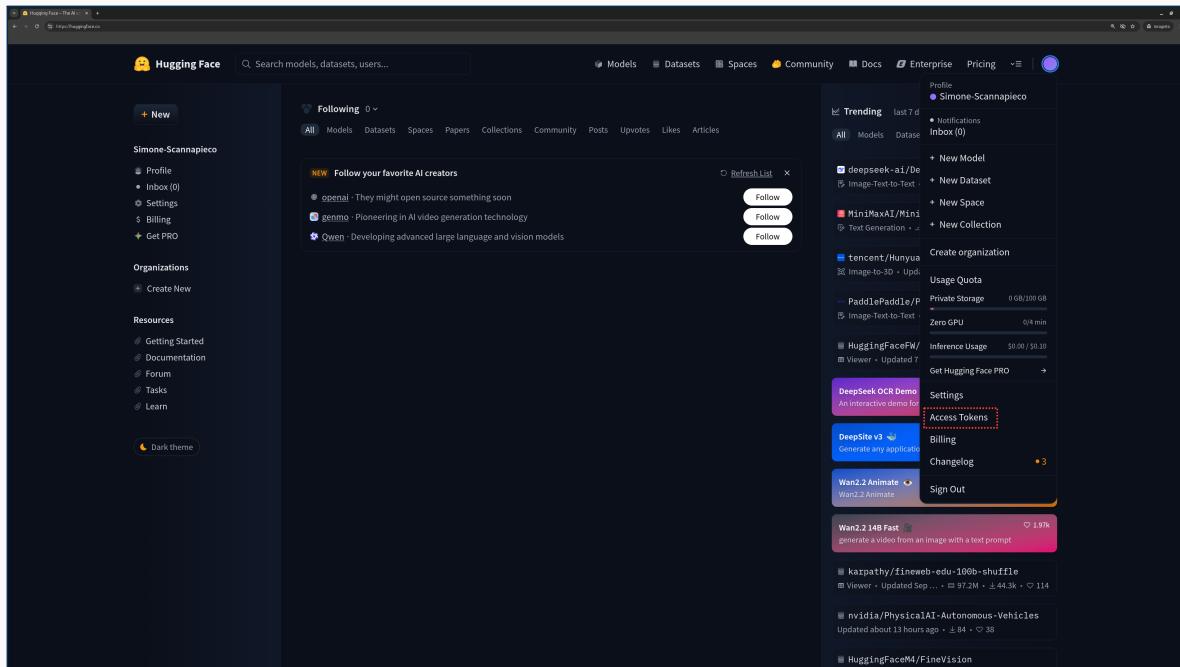
AMBIENTE DI SVILUPPO CREAZIONE ACCOUNT HUGGINGFACE

2 Creare una nuova utenza premendo il pulsante Sign up



Note

3 Accedere al portale e selezionare Access tokens nel menu in alto a destra



Note

AMBIENTE DI SVILUPPO CREAZIONE ACCOUNT HUGGINGFACE

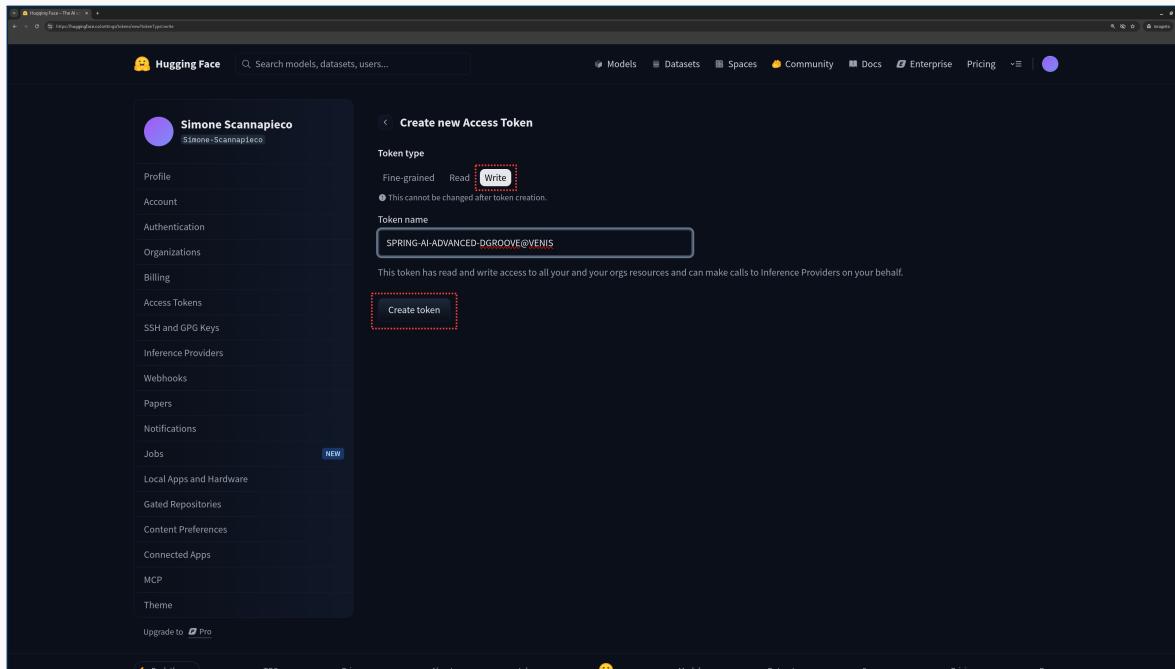
4 Premere sul pulsante Create new token

The screenshot shows the Hugging Face Hub profile page for user Simone Scannapieco. The left sidebar contains navigation links such as Profile, Account, Authentication, Organizations, Billing, Access Tokens (which is the current active section), SSH and GPG Keys, Inference Providers, Webhooks, Papers, Notifications, Jobs (with a NEW badge), Local Apps and Hardware, Gated Repositories, Content Preferences, Connected Apps, MCP, and Theme. At the bottom of the sidebar is an "Upgrade to Pro" button. The main content area is titled "Access Tokens" and contains a sub-section "User Access Tokens". It includes a note about tokens authenticating identity to the Hub and performing actions based on permissions, followed by a warning not to share them with anyone. A red box highlights the "+ Create new token" button. Below this is a table listing five access tokens with columns for Name, Value, Last Refreshed Date, Last Used Date, and Permissions (with options for WRITE, READ, or None).

Name	Value	Last Refreshed Date	Last Used Date	Permissions
» SPRING-AI-ADVANCED-DGROOVE@VENIS	hf_...HFYX	about 1 hour ago	-	WRITE
» ITS-LAST-COOKIE-2024	hf_...eCEp	13 days ago	Jan 9	READ
» CCAI_AIPERURANIUM	hf_...vHFU	Jul 23	-	READ
» SENTENCE-TRANSFORMERS-TEST-2025	hf_...KRav	May 26	May 26	READ
» ITS-LAST-SUSE-2024	hf_...mbQL	Dec 2, 2024	Dec 6, 2024	READ

Note

5 Selezionare token di tipo Write, denominare il token e premere Create token



 Simone Scannapieco

 Spring AI - Corso avanzato

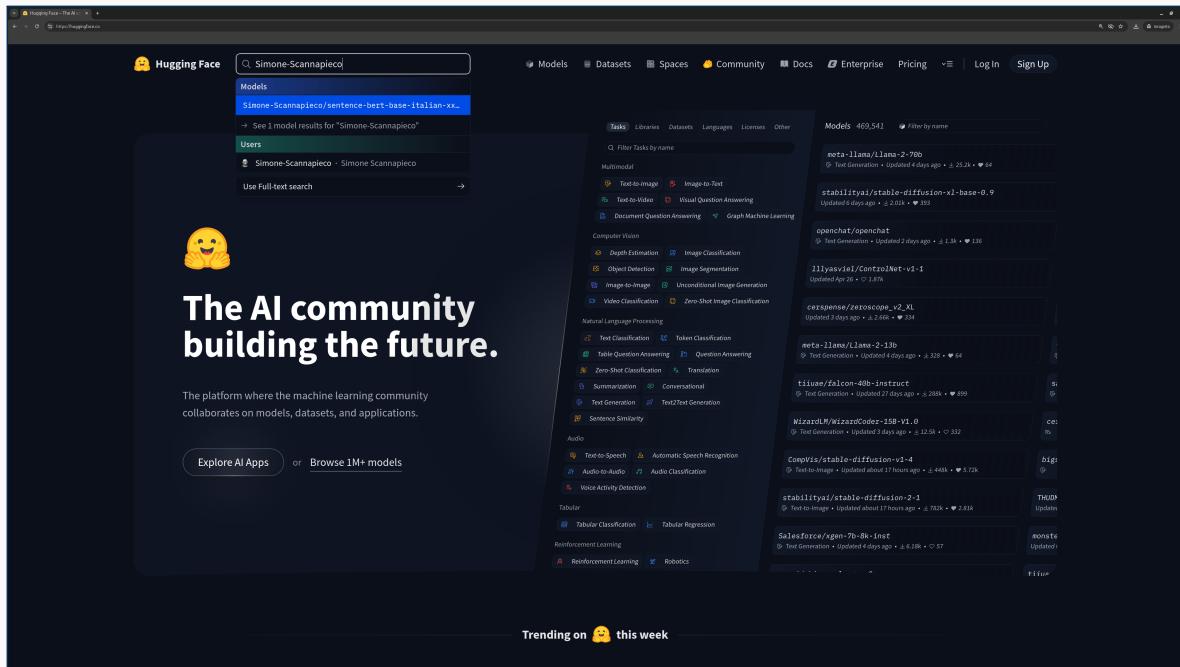
 Venis S.p.A, Venezia, IT

5 / 17

Note

AMBIENTE DI SVILUPPO CREAZIONE ACCOUNT HUGGINGFACE

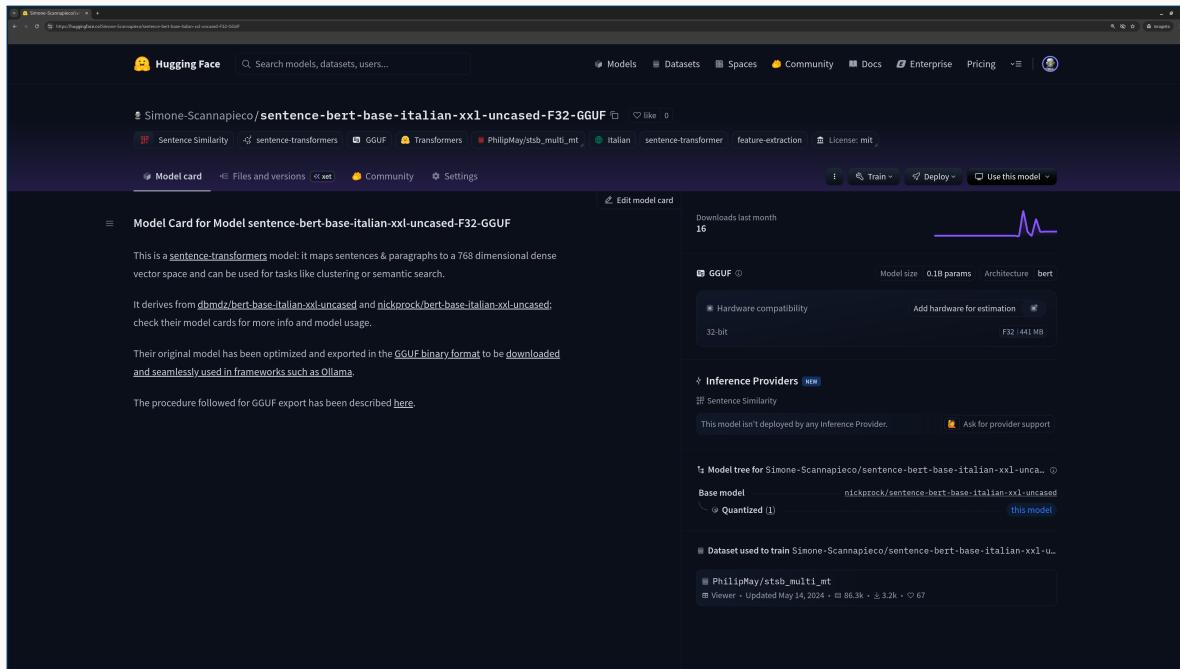
6 Cercare il modello di *embedding*, leggendone la card



Note

AMBIENTE DI SVILUPPO CREAZIONE ACCOUNT HUGGINGFACE

6 Cercare il modello di embedding, leggendone la card



Note

File launch.json

```
// Use IntelliSense to learn about possible attributes.  
// Hover to view descriptions of existing attributes.  
// For more information, visit: https://go.microsoft.com/fwlink/?linkid=830387  
"version": "0.2.0",  
"configurations": [  
    {  
        "type": "java",  
        "name": "Launch Current File",  
        "request": "launch",  
        "mainClass": "${file}"  
    },  
    {  
        "type": "java",  
        "name": "DemoApplication",  
        "request": "launch",  
        "mainClass": "it.venis.ai.spring.demo.DemoApplication",  
        "projectName": "demo",  
        "env": {  
            "GOOGLE_AI_API_KEY": "...",  
            "HUGGING_FACE_HUB_TOKEN": "..."  
        }  
    }  
]
```

Note

File settings.json

```
{
    "java.compile.nullAnalysis.mode": "disabled",
    "java.configuration.updateBuildConfiguration": "interactive",
    "java.test.config": {
        "env": {
            "GOOGLE_AI_API_KEY": "...",
            "HUGGING_FACE_HUB_TOKEN": "..."
        }
    }
}
```

Note

File docker-compose.yml

```
services:
  ...
  spring-ai-vector-store:
    image: qdrant/qdrant:${{QDRANT_VERSION:-latest}}
    hostname: spring-ai-vector-store
    container_name: spring_ai_vector_store
    ports:
      - ${{QDRANT_HTTP_PORT:-6333}}:6333
      - ${{QDRANT_GRPC_PORT:-6334}}:6334
    volumes:
      - spring_ai_vector_store:/qdrant/storage
    restart: unless-stopped

volumes:
  ...
  spring_ai_vector_store:
    name: spring_ai_vector_store
```

Note

File spring-ai.env

```
...  
# qdrant configuration  
QDRANT_VERSION=v1.13.0  
QDRANT_HTTP_PORT=6333  
QDRANT_GRPC_PORT=6334  
# default: latest  
# default: 6333  
# default: 6334
```

Note

Dipendenze di sistema aggiuntive

```
<dependency>
    <groupId>org.springframework.ai</groupId>
    <artifactId>spring-ai-rag</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.ai</groupId>
    <artifactId>spring-ai-advisors-vector-store</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.ai</groupId>
    <artifactId>spring-ai-starter-vector-store-qdrant</artifactId>
</dependency>
<!--
```

Note

Configurazione applicativo

```
spring:
  ...
  autoconfigure:
    exclude:
      - org.springframework.ai.vectorstore.qdrant.autoconfigure.QdrantVectorStoreAutoConfiguration
    # We must disable Vector Store auto-configuration because of two different EmbeddingModel beans
    # (OpenAI-Gemini and Ollama). The goal is to have two different vector collections, one for each
    # family of LLM.

ai:
  ollama:
    ...
    embedding:
      model: hf.co/Simone-Scannapieco/sentence-bert-base-italian-xxl-uncased-F32-GGUF

openai:
  ...
  embedding:
    options:
      model: gemini-embedding-001

vectorstore:
  qdrant:
    initialize-schema: true
    host: 172.17.0.1
    port: 6334
```

Note

File erase_llm_volumes.sh

```
#!/bin/bash

volume_name=spring_ai_llm

docker volume rm $volume_name
```

File erase_vector_store_volumes.sh

```
#!/bin/bash

volume_name=spring_ai_vector_store

docker volume rm $volume_name
```

Note

File get-rag-data-system-ita-prompt.st

Sei un assistente AI in grado di rispondere alle domande dell'utente solo in base al contesto fornito dalla sezione DOCUMENTI.

Se la risposta non è presente nella sezione DOCUMENTI, informa l'utente di non sapere la risposta.

DOCUMENTI: <documenti>

File get-rag-data-system-eng-prompt.st

You are an helpful assistant, answering questions based on the given context in the DOCUMENTS section and no prior knowledge.

If the answer is not in the DOCUMENTS section, then reply that you cannot answer to the question.

DOCUMENTS: <documenti>

Note

Configurazione Vector Store - I

```
package it.venis.ai.spring.demo.config;

import org.springframework.ai.embedding.EmbeddingModel;
import org.springframework.ai.openai.OpenAiEmbeddingModel;
import org.springframework.ai.vectorstore.VectorStore;
import org.springframework.ai.vectorstore.qdrant.QdrantVectorStore;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

import io.qdrant.client.QdrantClient;
import io.qdrant.client.QdrantGrpcClient;

@Configuration
public class RAGConfig {

    @Autowired
    public OpenAiEmbeddingModel openAiEmbeddingModel;

    @Autowired
    public EmbeddingModel ollamaEmbeddingModel;

    @Value("${spring.ai.vectorstore.qdrant.host:#{null}}")
    private String qdrantHost;
    @Value("${spring.ai.vectorstore.qdrant.port:#{null}}")
    private String qdrantPort;
    @Value("${spring.ai.vectorstore.qdrant.use-tls:#{null}}")
    private String useTls;
```

Note

Configurazione Vector Store - II

```
...
@Bean
public QdrantClient qdrantClient() {
    QdrantGrpcClient.Builder grpcClientBuilder = QdrantGrpcClient.newBuilder()
        .qdrantHost == null ? "localhost" : qdrantHost,
        .qdrantPort == null ? 6334 : Integer.valueOf(qdrantPort),
        .useTls == null ? false : Boolean.valueOf(useTls));
    return new QdrantClient(grpcClientBuilder.build());
}

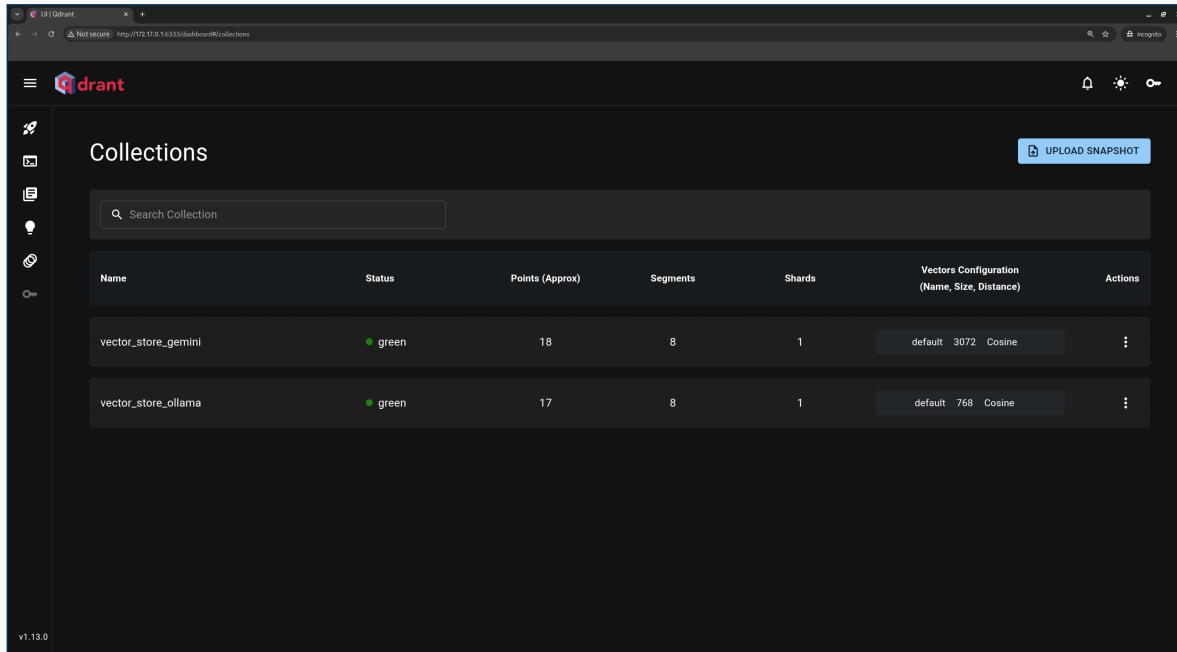
@Value("${spring.ai.vectorstore.qdrant.collection-name.gemini:#{null}}")
private String qdrantCollectionNameGemini;
@Value("${spring.ai.vectorstore.qdrant.collection-name.ollama:#{null}}")
private String qdrantCollectionNameOllama;
@Value("${spring.ai.vectorstore.qdrant.initialize-schema:#{null}}")
private String qdrantInitializeSchema;

@Bean
public VectorStore geminiVectorStore(QdrantClient qdrantClient,
    @Qualifier("openAiEmbeddingModel") EmbeddingModel geminiEmbeddingModel) {
    return QdrantVectorStore.builder(qdrantClient, geminiEmbeddingModel)
        .collectionName(qdrantCollectionNameGemini == null ? "vector_store_gemini" : qdrantCollectionNameGemini)
        .initializeSchema(qdrantInitializeSchema == null ? false : Boolean.valueOf(qdrantInitializeSchema))
        .build();
}

@Bean
public VectorStore ollamaVectorStore(QdrantClient qdrantClient,
    @Qualifier("ollamaEmbeddingModel") EmbeddingModel ollamaEmbeddingModel) {
    return QdrantVectorStore.builder(qdrantClient, ollamaEmbeddingModel)
        .collectionName(qdrantCollectionNameOllama == null ? "vector_store_ollama" : qdrantCollectionNameOllama)
        .initializeSchema(qdrantInitializeSchema == null ? false : Boolean.valueOf(qdrantInitializeSchema))
        .build();
}
```

Note

1 Verificare la dashboard Qdrant (<http://172.17.0.1:6333/dashboard>)



Note

<https://github.com/simonescannapieco/spring-ai-advanced-dgroove-venis-code.git>
Branch: 7-spring-ai-gemini-ollama-rag-text-to-vector-store

Note