dgroove
Tecnologia emozionale.

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

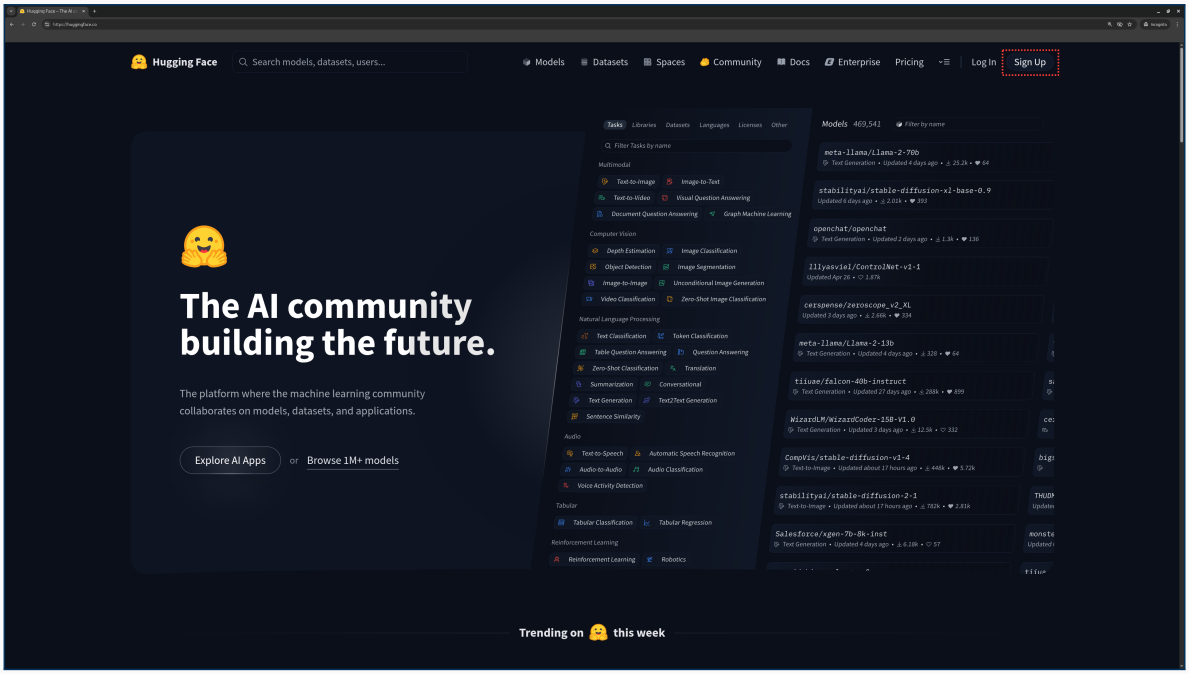
- Storico della conversazione valida per singola sessione

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

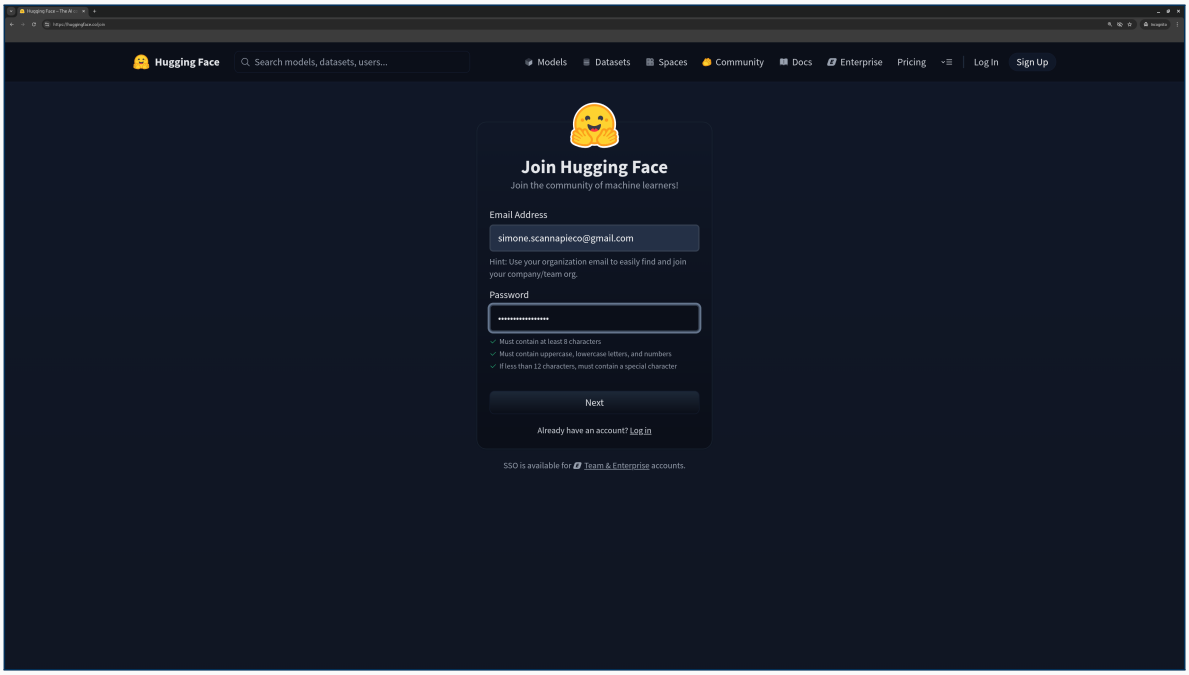
Note

1. Accedere al portale `https://huggingface.co/`



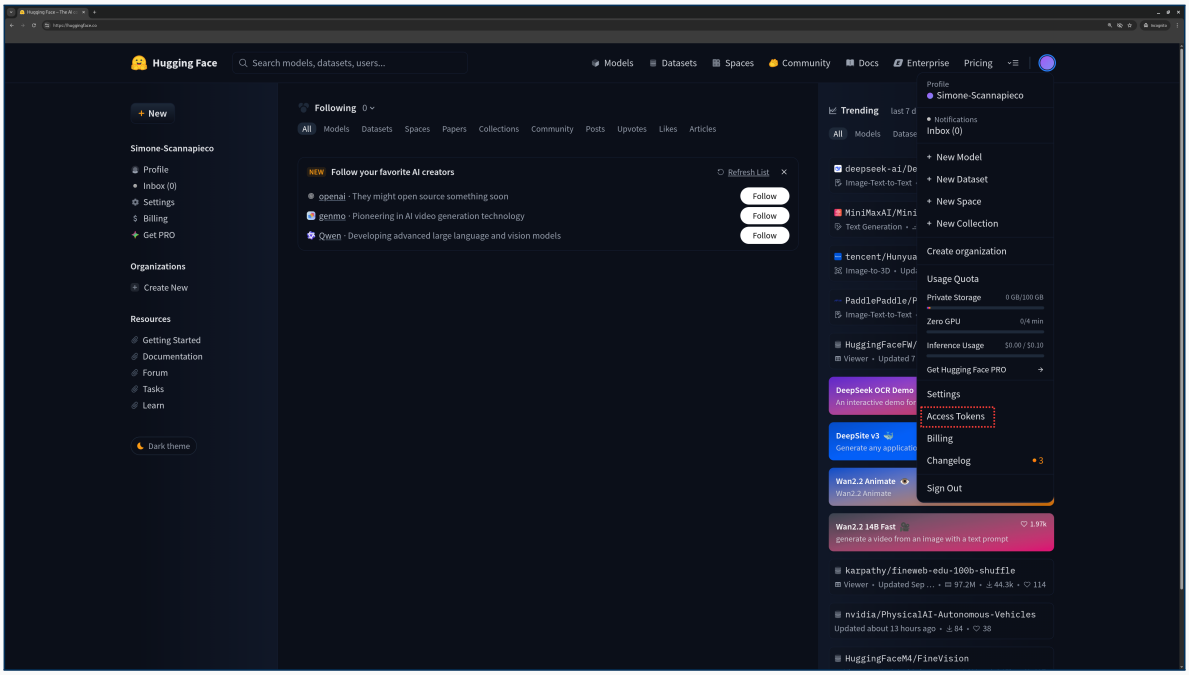
Note

2. Creare una nuova utenza premendo il pulsante `Sign up`



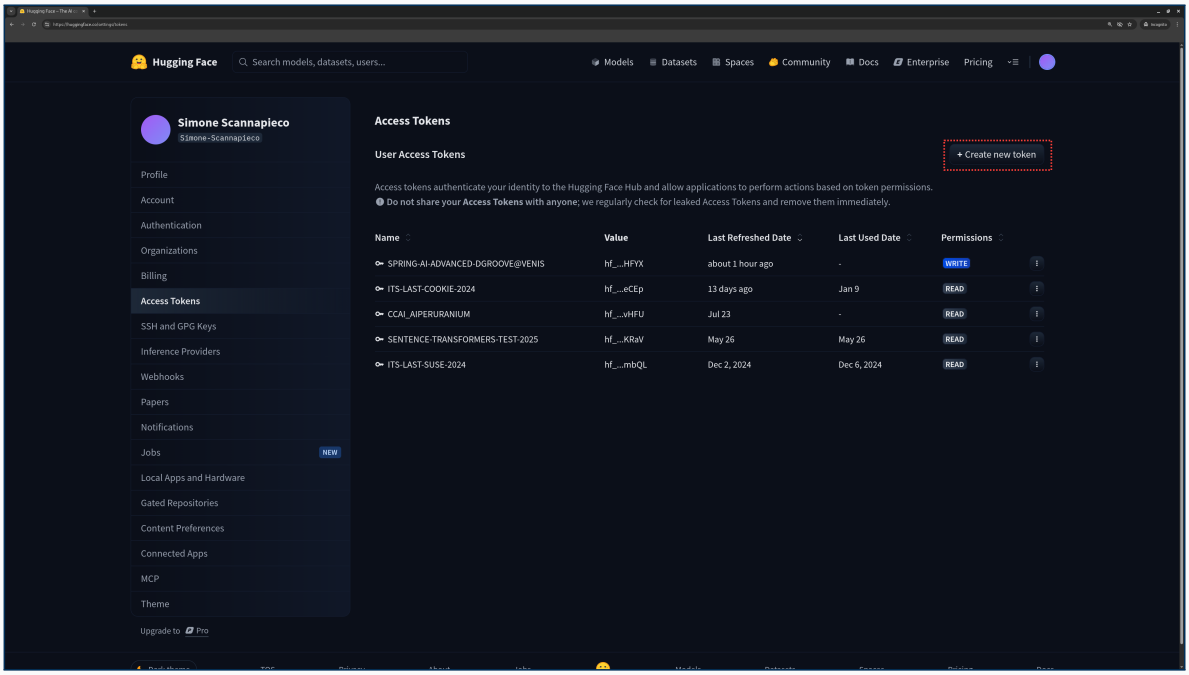
Note

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



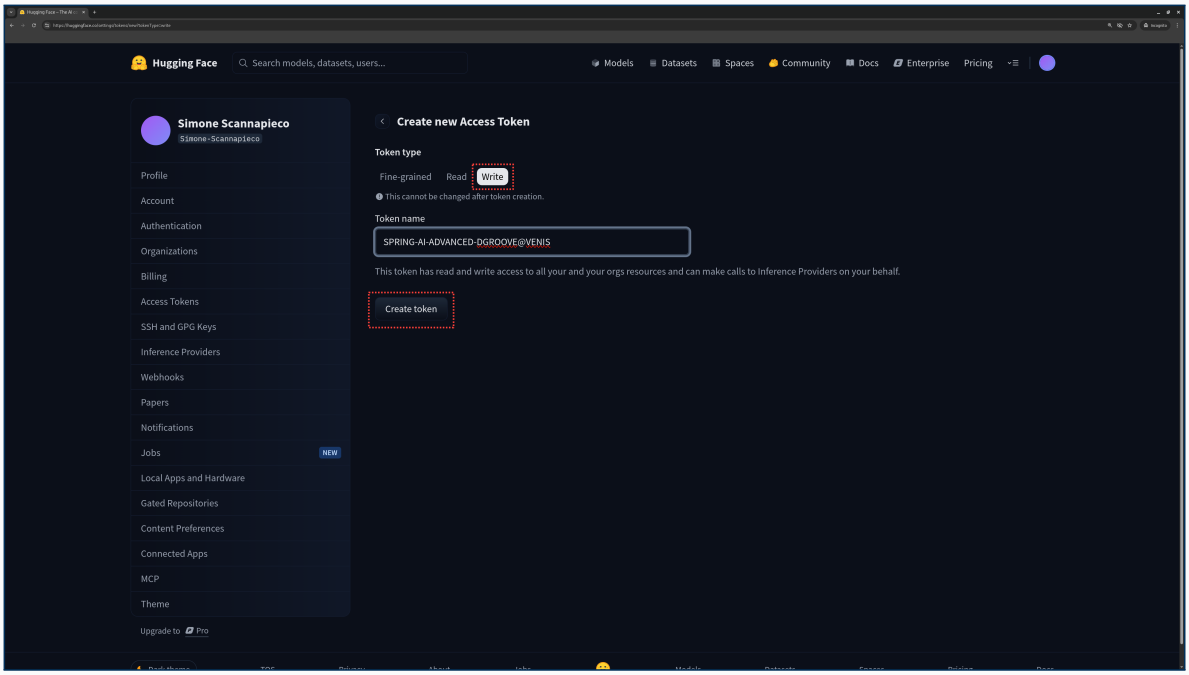
Note

4 Premere sul pulsante `Create new token`



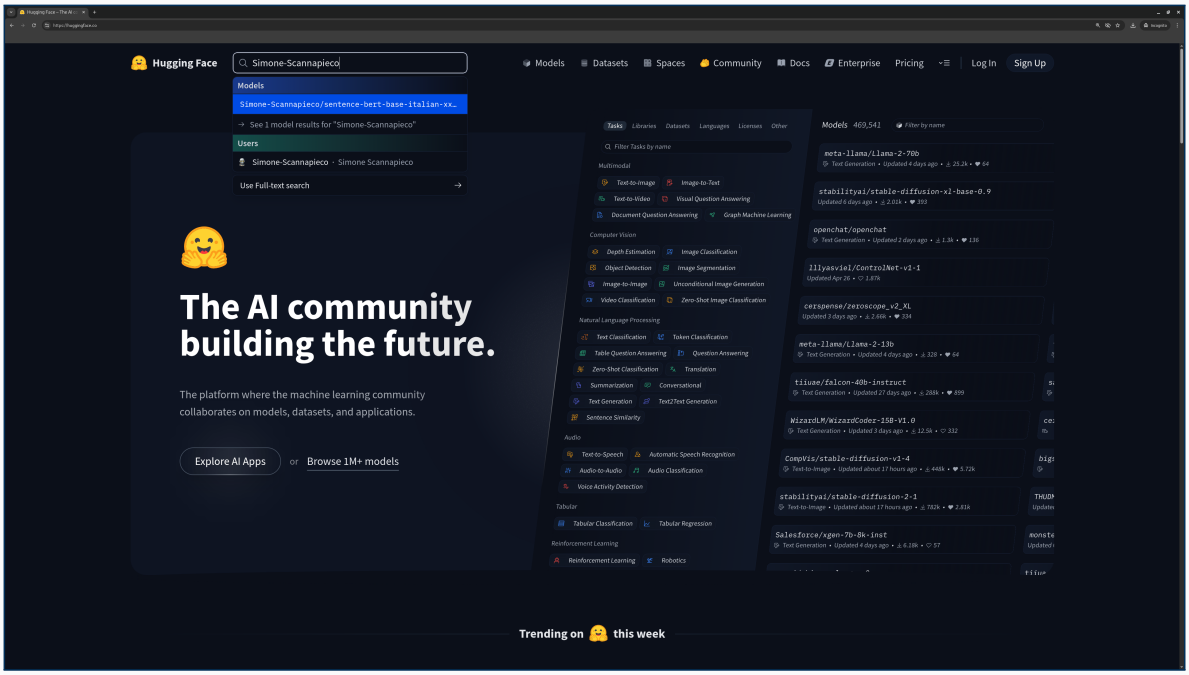
Note

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



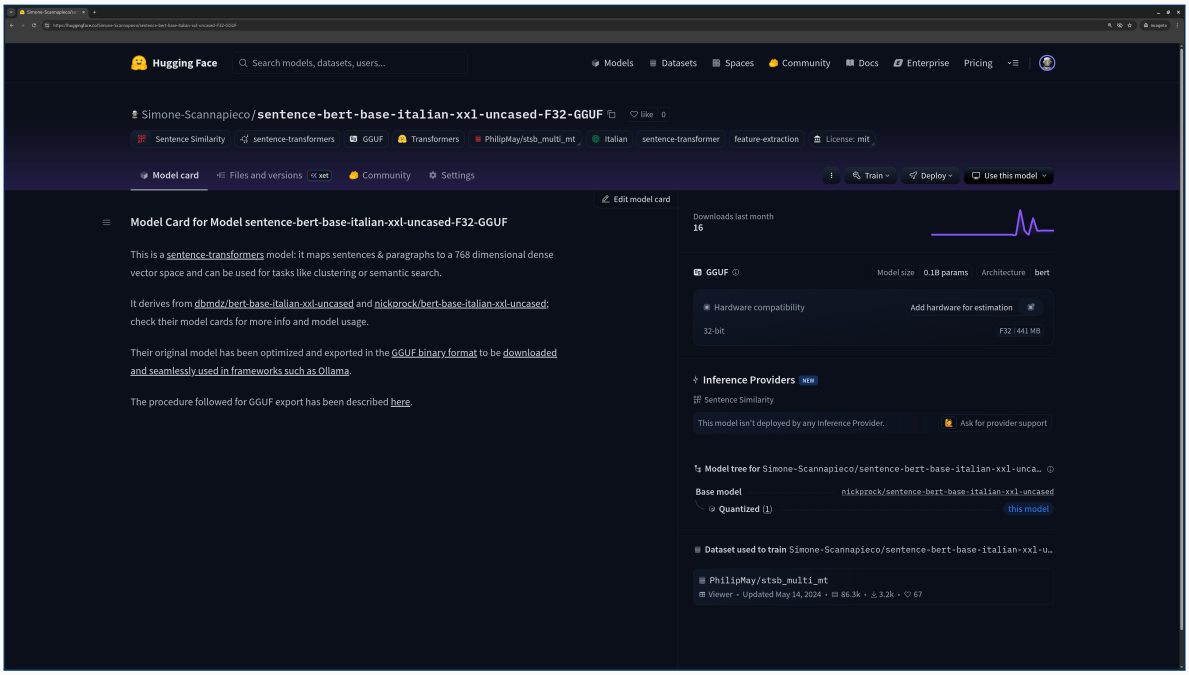
Note

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



Note

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`

```yaml
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_GRCP_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                                  # default: latest
QDRANT_HTTP_PORT=6333                                     # default: 6333
QDRANT_GRCP_PORT=6334                                     # 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

```yaml
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

Componente popolamento Qdrant

```java
package it.venis.ai.spring.demo.rag;

import java.util.List;
import java.util.stream.Collectors;

import org.springframework.ai.document.Document;
import org.springframework.ai.vectorstore.SearchRequest;
import org.springframework.ai.vectorstore.VectorStore;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.stereotype.Component;

import jakarta.annotation.PostConstruct;

@Component
public class TextDataLoader {

    private final VectorStore geminiVectorStore;
    private final VectorStore ollamaVectorStore;

    public TextDataLoader(@Qualifier("geminiVectorStore") VectorStore geminiVectorStore,
            @Qualifier("ollamaVectorStore") VectorStore ollamaVectorStore) {
        this.geminiVectorStore = geminiVectorStore;
        this.ollamaVectorStore = ollamaVectorStore;

    }

    @PostConstruct
    public void loadVenisInfoIntoVectorStore() {
        List<String> venisInfo = List.of(
                "VENIS SpA, acronym for Venice Informatics and Systems, is the ICT services company and local electronic communications

                "The legal and main operational headquarters of VENIS SpA is located in the prestigious Palazzo Ziani, situated in Sesti

                "The corporate structure of VENIS SpA provides for a share capital of 1,549,500 euros fully paid up, divided into 30,000

                "From an economic-financial perspective, VENIS SpA recorded in 2023 a turnover of 21,685,009 euros, confirming its solid

                "The corporate mission of VENIS SpA is to bring innovation to serve the city and support the Municipality of Venice, its

                "The corporate purpose of VENIS SpA includes the design, implementation, installation and operational management of info

                "The main services offered by VENIS SpA include the development and management of the municipal information system to im

                "The Smart Control Room project represents the flagship of VENIS SpA's technological innovation for the city of Venice.

                "The technological architecture of the Smart Control Room is based on cutting-edge technologies such as artificial intel

                "The Smart Control Room operations involve synergistic collaboration of about thirty operators from different institutio

                "The Smart City 2.0 model implemented by VENIS SpA in Venice represents an innovative paradigm that surpasses the tradit

                "The open data approach adopted by VENIS SpA ensures that data collected by the Smart Control Room, appropriately anonym

                "The results achieved by VENIS SpA in the Smart Cities field have been recognized at national and international levels.

                "The network infrastructures managed by VENIS SpA constitute the backbone of city connectivity. The company has created

                "VENIS SpA's commitment to training and technology transfer is realized through continuous training programs for municip

                "VENIS SpA's recruitment strategies reflect the continuous need for specialized skills to support technological innovati

                "The VENIS SpA e-procurement platform represents an example of digitalization of public purchasing processes. The system

                "VENIS SpA's future vision is oriented towards the further evolution of the Smart City model, with particular attention
        );
        SearchRequest searchRequest = SearchRequest.builder()
                .query("Check")
                .similarityThresholdAll()
                .build();

        List<Document> similarDocs = geminiVectorStore.similaritySearch(searchRequest);

        if (similarDocs.size() == 0) {

            List<Document> documents =
            venisInfo.stream().map(Document::new).collect(Collectors.toList());
            this.geminiVectorStore.add(documents);

        }

    }

    @PostConstruct
    public void loadSSCVInfoIntoVectorStore() {
        List<String> ssCVInfo = List.of(
                "Simone Scannapieco è un professionista italiano nato il 3 febbraio 1982 a Verona, dove attualmente risiede in Via Amede
```


Note

Interfaccia servizio

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

import it.venis.ai.spring.demo.model.Answer;
import it.venis.ai.spring.demo.model.QuestionRequest;

public interface RAGService {

    public Answer getGeminiRAGTextToVectorStoreAnswer(QuestionRequest request);

    public Answer getOllamaRAGTextToVectorStoreAnswer(QuestionRequest request);

}
```


Note

Implementazione servizio

```java
package it.venis.ai.spring.demo.services;

import java.util.List;
import java.util.Map;
import java.util.stream.Collectors;

import org.springframework.ai.chat.client.ChatClient;
import org.springframework.ai.chat.memory.ChatMemory;
import org.springframework.ai.document.Document;
import org.springframework.ai.template.st.StTemplateRenderer;
import org.springframework.ai.vectorstore.SearchRequest;
import org.springframework.ai.vectorstore.VectorStore;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Configuration;
import org.springframework.core.io.Resource;
import org.springframework.stereotype.Service;

import it.venis.ai.spring.demo.model.Answer;
import it.venis.ai.spring.demo.model.QuestionRequest;

@Service
@Configuration
public class RAGServiceImpl implements RAGService {

    private final ChatClient geminiChatClient;
    private final ChatClient ollamaChatClient;
    private final ChatClient ollamaMemoryChatClient;
    private VectorStore geminiVectorStore;
    private VectorStore ollamaVectorStore;

    public RAGServiceImpl(
            @Qualifier("geminiChatClient") ChatClient geminiChatClient,
            @Qualifier("ollamaChatClient") ChatClient ollamaChatClient,
            @Qualifier("ollamaMemoryChatClient") ChatClient ollamaMemoryChatClient,
            @Qualifier("geminiVectorStore") VectorStore geminiVectorStore,
            @Qualifier("ollamaVectorStore") VectorStore ollamaVectorStore) {

        this.geminiChatClient = geminiChatClient;
        this.ollamaChatClient = ollamaChatClient;
        this.ollamaMemoryChatClient = ollamaMemoryChatClient;
        this.geminiVectorStore = geminiVectorStore;
        this.ollamaVectorStore = ollamaVectorStore;

    }

    @Value("classpath:templates/get-rag-data-system-eng-prompt.st")
    private Resource ragDataSystemEngPrompt;

    @Override
    public Answer getGeminiRAGTextToVectorStoreAnswer(QuestionRequest request) {

        SearchRequest searchRequest = SearchRequest.builder()
                .query(request.body().question())
                .topK(4)
                .similarityThreshold(.2)
                .build();

        List<Document> similarDocs = geminiVectorStore.similaritySearch(searchRequest);

        String similarDocsString = similarDocs.stream()
                .map(Document::getText)
                .collect(Collectors.joining(System.lineSeparator()));

        return new Answer(this.geminiChatClient.prompt()
                // .advisors(advisorSpec -> advisorSpec.param(ChatMemory.CONVERSATION_ID,
                // request.username()))
                .system(s -> s.text(this.ragDataSystemEngPrompt)
                        .params(Map.of("documenti", similarDocsString)))
                .user(request.body().question())
                .templateRenderer(StTemplateRenderer.builder().startDelimiterToken('<')
                        .endDelimiterToken('>')
                        .build())
                .call()
                .content());
    }

    @Value("classpath:templates/get-rag-data-system-ita-prompt.st")
    private Resource ragDataSystemItaPrompt;

    @Override
    public Answer getOllamaRAGTextToVectorStoreAnswer(QuestionRequest request) {

        SearchRequest searchRequest = SearchRequest.builder()
                .query(request.body().question())
                .topK(4)
                .similarityThreshold(.3)
                .build();
```

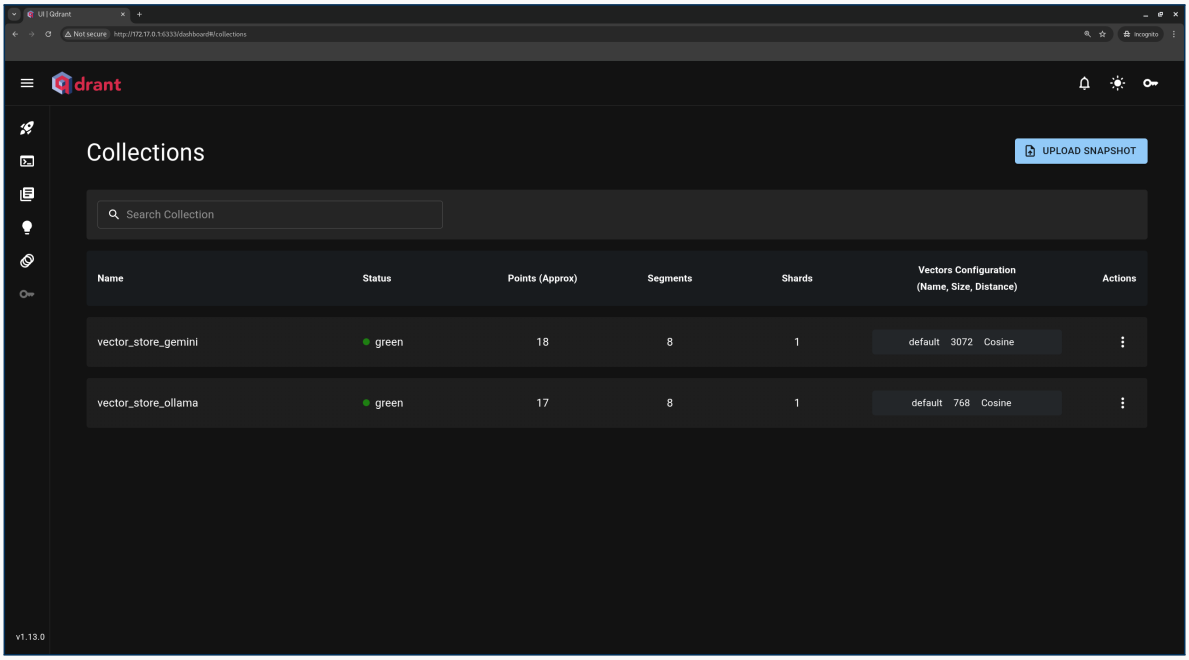

Note

Implementazione controllore REST

```java
package it.venis.ai.spring.demo.controllers;

import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;

import it.venis.ai.spring.demo.model.Answer;
import it.venis.ai.spring.demo.model.Question;
import it.venis.ai.spring.demo.model.QuestionRequest;
import it.venis.ai.spring.demo.services.QuestionService;
import it.venis.ai.spring.demo.services.RAGService;

@RestController
public class QuestionController {

    private final QuestionService service;
    private final RAGService ragService;

    public QuestionController(QuestionService service, RAGService ragService) {

        this.service = service;
        this.ragService = ragService;

    }

    @PostMapping("/gemini/ask")
    public Answer geminiAskQuestion(@RequestBody Question question) {

        return this.service.getGeminiAnswer(question);

    }

    @PostMapping("/ollama/ask")
    public Answer ollamaAskQuestion(@RequestBody Question question) {

        return this.service.getOllamaAnswer(question);

    }

    @PostMapping("/ollama/ask/default")
    public Answer ollamaAskDefaultQuestion(@RequestBody Question question) {

        return this.service.getOllamaDefaultAnswer(question);

    }

    @PostMapping("/ollama/ask/memory")
    public Answer getOllamaMemoryAwareAnswer(@RequestBody Question question) {

        return this.service.getOllamaMemoryAwareAnswer(question);

    }

    @PostMapping("/ollama/ask/memory/user")
    public Answer getOllamaPerUserMemoryAwareAnswer(@RequestBody QuestionRequest request) {

        return this.service.getOllamaPerUserMemoryAwareAnswer(request);

    }

    @PostMapping("/gemini/ask/rag/text-to-vs/venis")
    public Answer getGeminiRAGTextToVectorStoreAnswer(@RequestBody QuestionRequest request) {

        return this.ragService.getGeminiRAGTextToVectorStoreAnswer(request);

    }

    @PostMapping("/ollama/ask/rag/text-to-vs/cv")
    public Answer getOllamaRAGTextToVectorStoreAnswer(@RequestBody QuestionRequest request) {

        return this.ragService.getOllamaRAGTextToVectorStoreAnswer(request);

    }

}
```


Note

1. Verificare la *dashboard* Qdrant (`http://172.17.0.1:6333/dashboard`)



Note

Codice
Branch di riferimento

https://github.com/simonescannapieco/spring-ai-advanced-dgroove-venis-code.git

Branch: 7-spring-ai-gemini-ollama-rag-text-to-vector-store

Note