



# 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

## → Chatbot Ollama-CV e Gemini-Venis

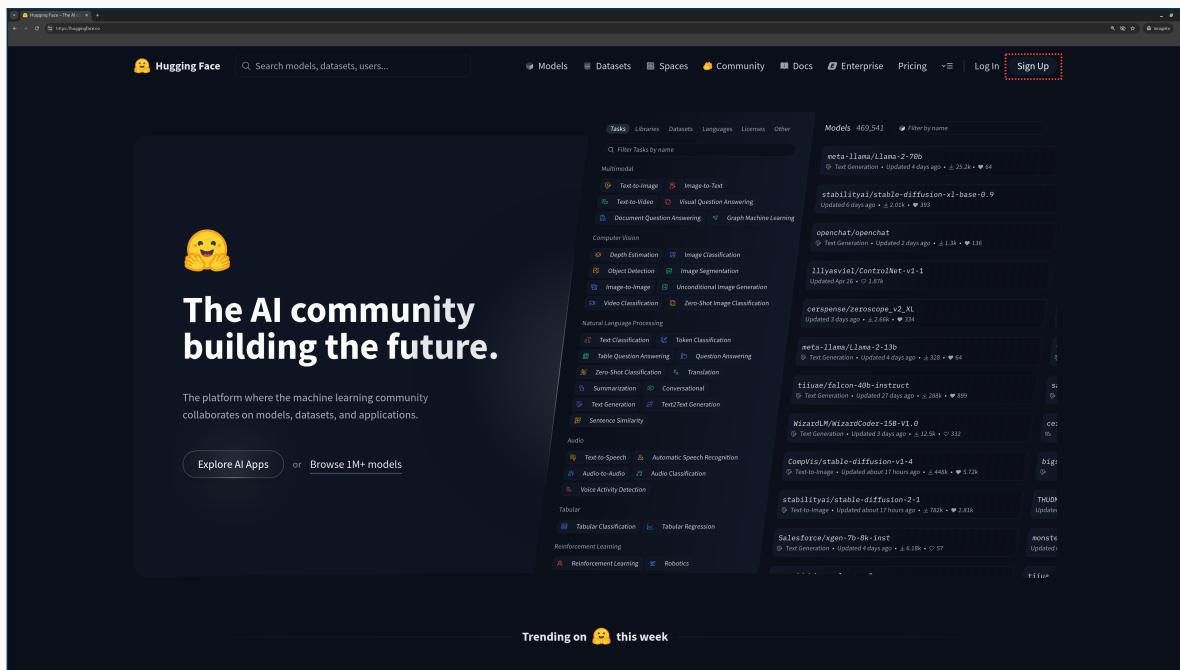
**A** Approccio naive (informazione testuale estratta tramite GENAI-assisted web scraping/summarization)

- 1 Isrizione al portale HuggingFace e creazione access token
  - 2 Verifica e modifica variabili di ambiente
  - 3 Creazione configurazione ambiente Docker Qdrant
  - 4 Modifica dipendenze di progetto
  - 5 Modifica configurazione applicativo
  - 6 Creazione *prompt templates* per strategia RAG
  - 7 Configurazione vector store per Ollama e Gemini *embeddings*
  - 8 Creazione component per popolamento vector store (dati Venis e CV)
  - 9 Creazione interfaccia e implementazione del servizio
  - 10 Modifica controllore MVC
  - 11 Test delle funzionalità con Postman/Insomnia

## Note

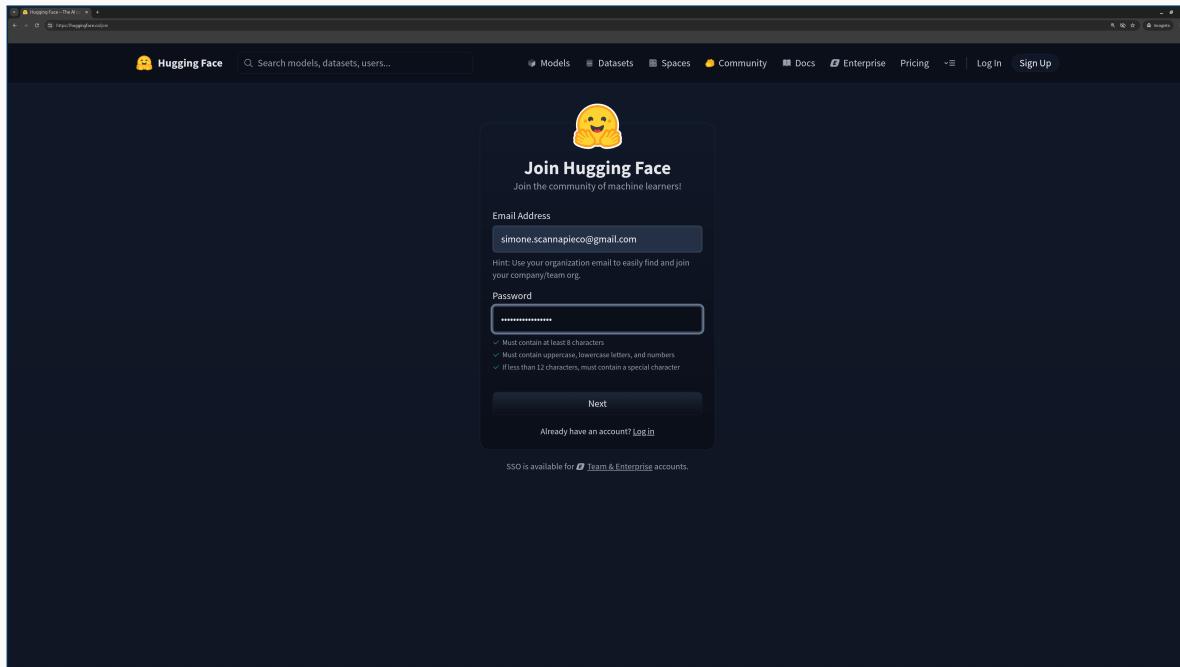
## **AMBIENTE DI SVILUPPO CREAZIONE ACCOUNT HUGGINGFACE**

1 Accedere al portale <https://huggingface.co/> e cliccare il pulsante Sign up



## Note

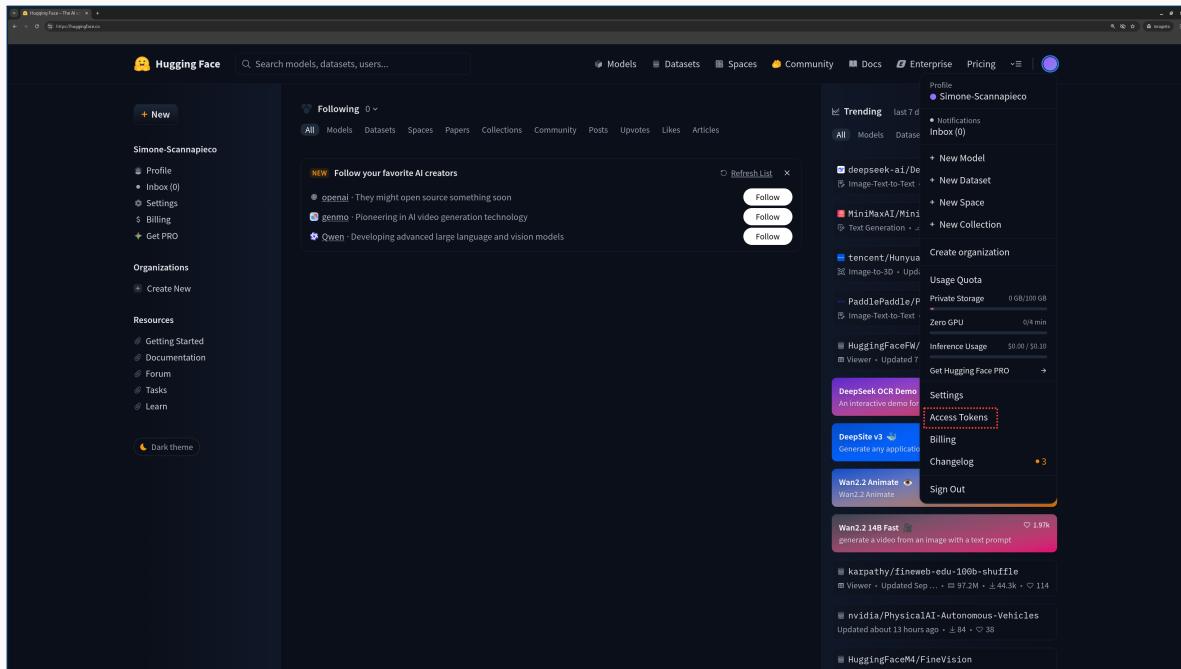
## 2 Creare una nuova utenza



## Note

## **AMBIENTE DI SVILUPPO CREAZIONE ACCOUNT HUGGINGFACE**

**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

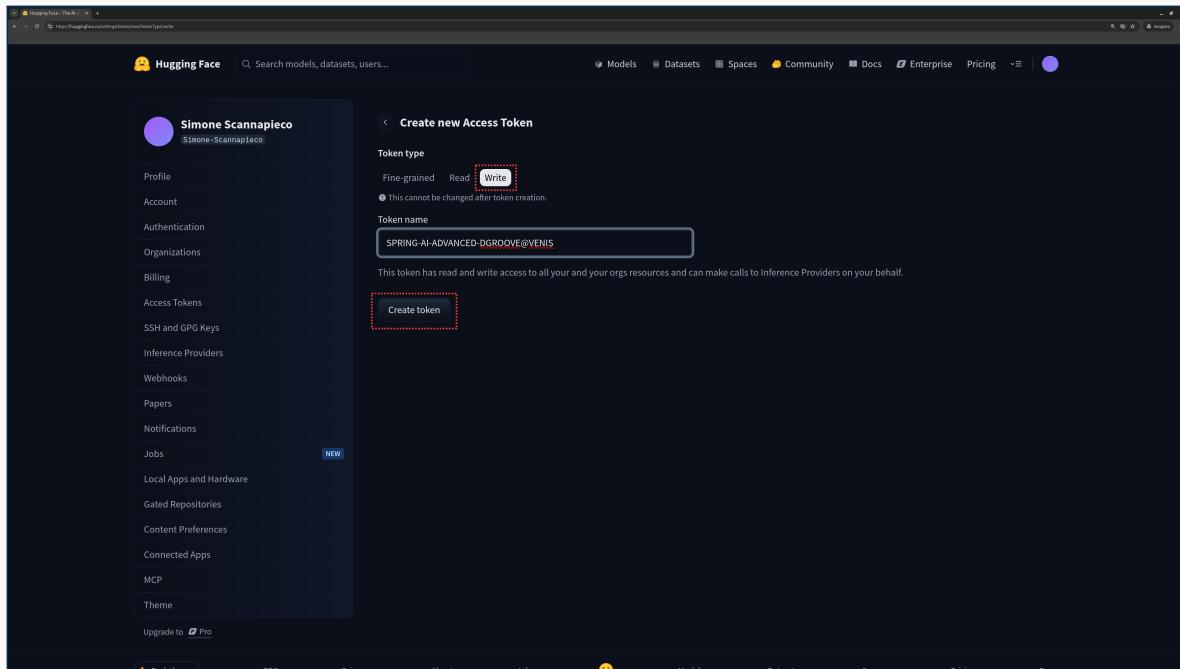
The screenshot shows the Hugging Face Hub profile page for user Simone Scannapieco. The left sidebar contains navigation links for Profile, Account, Authentication, Organizations, Billing, Access Tokens (which is the active tab), SSH and GPG Keys, Inference Providers, Webhooks, Papers, Notifications,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

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

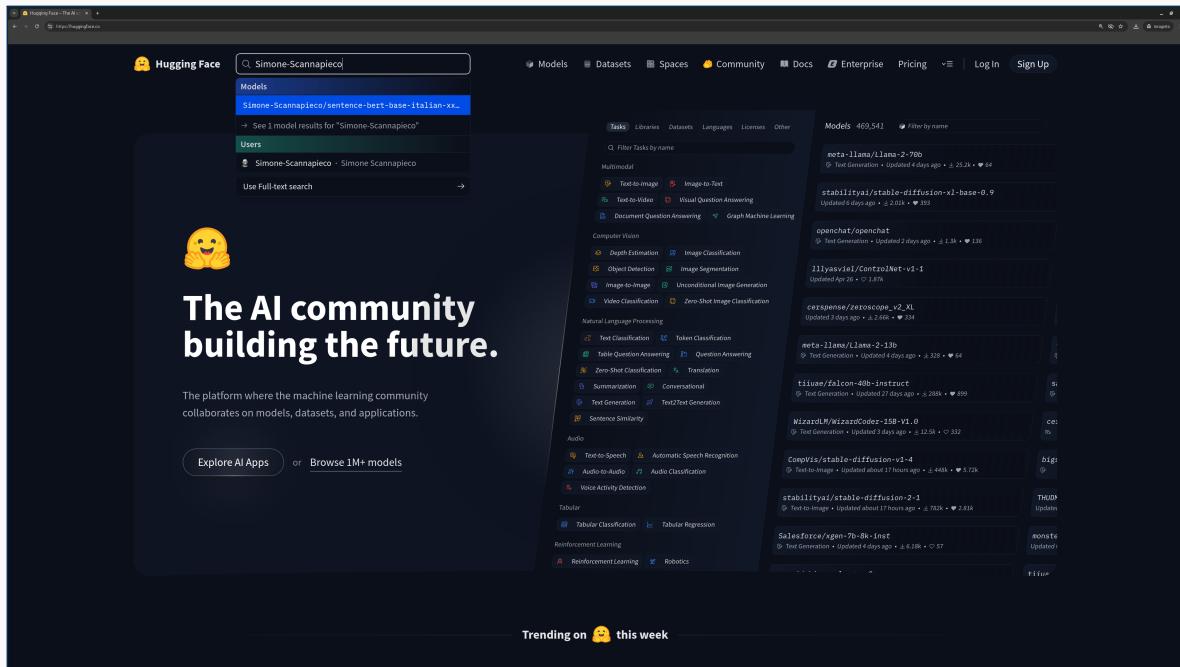
## **AMBIENTE DI SVILUPPO CREAZIONE ACCOUNT HUGGINGFACE**

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



## Note

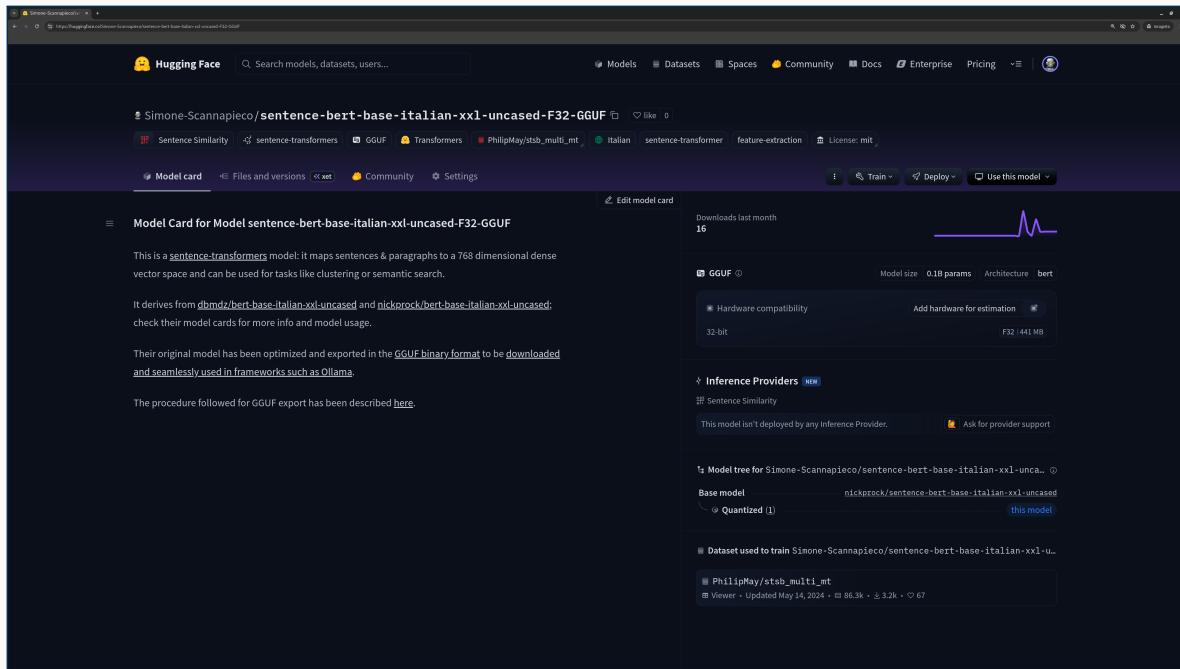
## 6 Cercare il modello di embedding, leggendo la card



## 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

## Componente popolamento Qdrant - I

```
package it.venis.ai.spring.demo.rag;
...
@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(...);
        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);
        }
    }
}
```

## Note

## Componente popolamento Qdrant - II

```
...
@PostConstruct
public void loadSSCVInfoIntoVectorStore() {
    List<String> ssCVInfo = List.of(...);
    SearchRequest searchRequest = SearchRequest.builder()
        .query("Check")
        .similarityThresholdAll()
        .build();
    List<Document> similarDocs = ollamaVectorStore.similaritySearch(searchRequest);
    if (similarDocs.size() == 0) {
        List<Document> documents = ssCVInfo.stream().map(Document::new).collect(Collectors.toList());
        this.ollamaVectorStore.add(documents);
    }
}
```

## 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 - I**

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

...
@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;
    }

}
```

## Note

## **Implementazione servizio - II**

```
...
@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()
        .system(s -> s.text(this.ragDataSystemEngPrompt)
            .params(Map.of("documenti", similarDocsString)))
        .user(request.body().question())
        .templateRenderer(StTemplateRenderer.builder().startDelimiterToken('<')
            .endDelimiterToken('>')
            .build())
        .call()
        .content());
}

...
```

## Note

## **Implementazione servizio - III**

```
...  
@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();  
  
    List<Document> similarDocs = ollamaVectorStore.similaritySearch(searchRequest);  
  
    String similarDocsString = similarDocs.stream()  
        .map(Document::getText)  
        .collect(Collectors.joining(System.lineSeparator()));  
  
    return new Answer(this.ollamaMemoryChatClient.prompt()  
        .advisors(advisorSpec -> advisorSpec.param(ChatMemory.CONVERSATION_ID, request.username()))  
        .system(s -> s.text(this.ragDataSystemItaPrompt)  
            .params(Map.of("documenti", similarDocsString)))  
        .user(request.body().question())  
        .templateRenderer(StTemplateRenderer.builder().startDelimiterToken('<')  
            .endDelimiterToken('>')  
            .build())  
        .call()  
        .content());  
}  
}
```

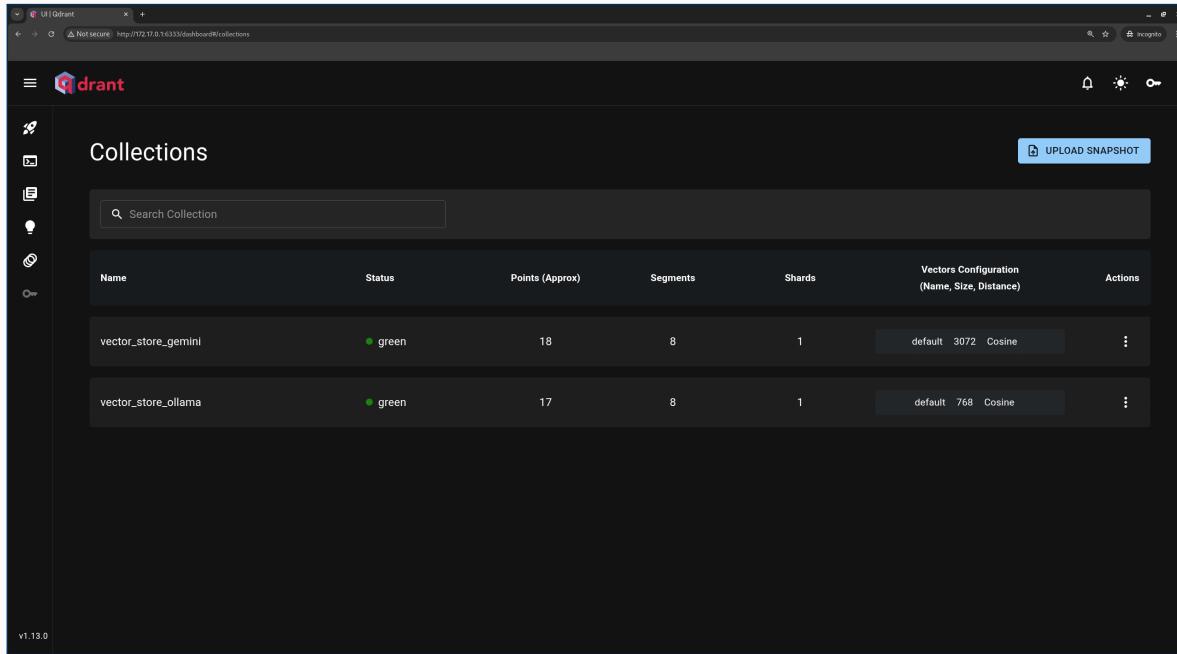
## Note

# Implementazione controllore REST

```
package it.venis.ai.spring.demo.controllers;  
...  
@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/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

**⚠️ 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