

Informe de la Pràctica: Java Servlets i Chatbot RAG

1. Introducció i Entorn de la Pràctica

1.1 Objectius de la Pràctica

Aquesta pràctica té com a objectiu desenvolupar una aplicació web completa utilitzant Java Servlets amb Apache Tomcat 10, implementant un sistema de lloguer de vehicles amb dues funcionalitats principals:

- Gestió de nous lloguers de vehicles
- Visualització de la llista de lloguers (amb autenticació d'administrador)

A més, s'ha implementat una extensió que consisteix en un chatbot intel·ligent utilitzant tècniques de Retrieval-Augmented Generation (RAG) per millorar l'experiència de l'usuari.

1.2 Entorn Tecnològic

- **Servidor web:** Apache Tomcat 10.0.10
- **Llenguatge de programació:** Java (servlets)
- **Format de dades:** JSON per a la persistència
- **Containerització:** Docker
- **IA:** Ollama amb models Llama3.2 i Llama3.1
- **Framework RAG:** LangChain amb embeddings locals

2. Alternatives per Resoldre la Pràctica

2.1 Alternatives per al Backend Web

Opció 1: Java Servlets (Solució triada)

- Avantatges: Integració nativa amb Java, robustesa, escalabilitat
- Desavantatges: Codi més verbós, configuració complexa

2.2 Alternatives per a la Persistència de Dades

Opció 1: Fitxer JSON (Solució triada)

- Avantatges: Simplicitat, fàcil de debugar
- Desavantatges: No escalable, problemes de concorrència

Opció 2: Base de dades MySQL

- Avantatges: Escalabilitat, transaccions ACID
- Desavantatges: Més complexitat de configuració

Opció 3: Base de dades en memòria

- Avantatges: Ràpida
- Desavantatges: Pèrdua de dades al reiniciar

2.3 Alternatives per al Chatbot

Opció 1: RAG amb Ollama (Solució triada)

- Avantatges: Privacitat, control total, cost zero
- Desavantatges: Requereix recursos locals

Opció 2: API externa (OpenAI, Anthropic)

- Avantatges: Millor qualitat, menys configuració
- Desavantatges: Cost, dependència externa, privacitat

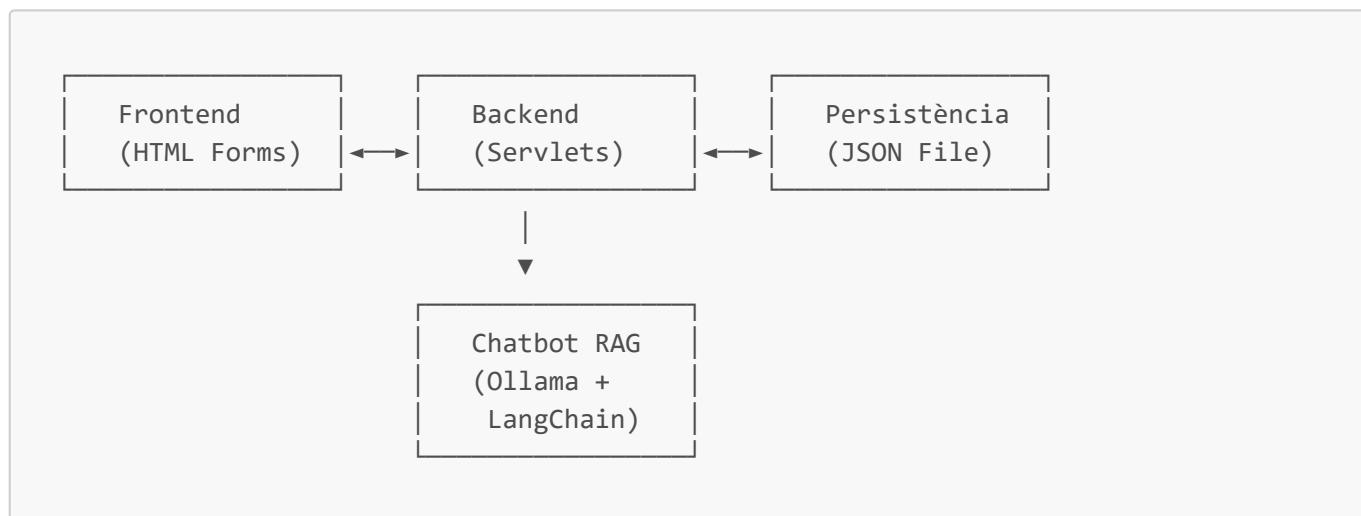
Opció 3: Chatbot basat en regles

- Avantatges: Predictible, ràpid
- Desavantatges: Limitada flexibilitat

3. Solució Triada i Implementació

3.1 Arquitectura de la Solució

La solució implementada segueix la següent arquitectura:



3.2 Implementació dels Servlets

3.2.1 Servlet CarRentalNew.java

El servlet **CarRentalNew** gestiona la creació de nous lloguers. Les característiques clau inclouen:

- Gestió robusta d'errors amb try-catch
- Creació automàtica del fitxer JSON si no existeix
- Reutilització de dades existents
- Interfície HTML dinàmica

```

public void handleCreateRental(String co2Rating, String engine, String
dias_alquiler,
                                String num_vehi, Double descuento, PrintWriter out)
  
```

```
{
    // Gestió del fitxer JSON
    // Creació/actualització de lloguers
    // Persistència a disc
}
```

3.2.2 Servlet CarRentalList.java

El servlet **CarRentalList** gestiona la visualització de lloguers amb:

- Autenticació d'administrador (admin/1234)
- Compatibilitat amb diferents formats de dades

```
public void handleReadRental(HttpServletRequest res) {
    // Gestió fitxer JSON (parse)
    // Itera per rental array i imprimeix cada rental detail
    // Controla els errors en llegir el JSON
}
```

3.3 Implementació del Chatbot RAG

3.3.1 Arquitectura RAG

El sistema RAG implementat segueix el següent flux de treball:

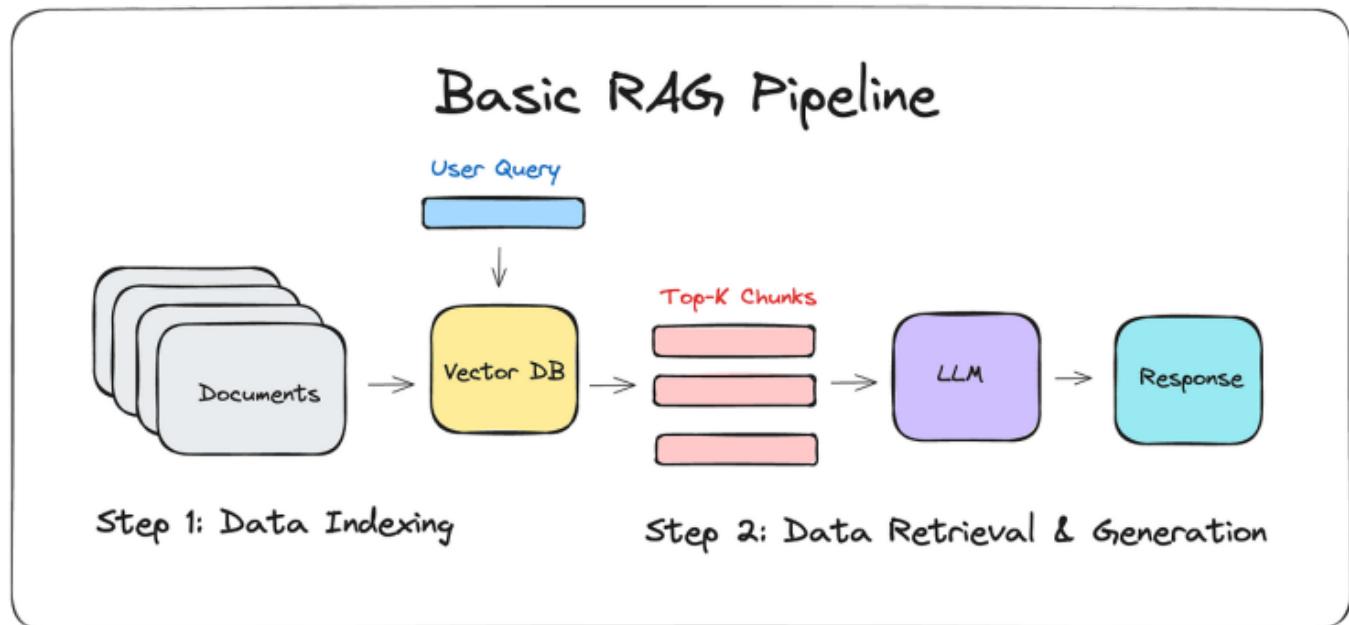


Figura 1: Pipeline RAG per al sistema de chatbot

3.3.2 Creació de l'Índex Vectorial

El sistema utilitza LangChain per crear un índex vectorial del dataset de lloguers:

```
def create_vector_index(model_name="llama3.2"):  
    # Carregar dataset CSV  
    loader = TextLoader("dataset/rentals.csv")  
  
    # Configurar embeddings  
    embeddings = OllamaEmbeddings(model=model_name)  
  
    # Crear índex vectorial  
    index_creator = VectorstoreIndexCreator(embedding=embeddings)  
    index = index_creator.from_loaders([loader])
```

3.3.3 Model d'Embeddings

El sistema utilitza embeddings per representar el coneixement de manera vectorial:

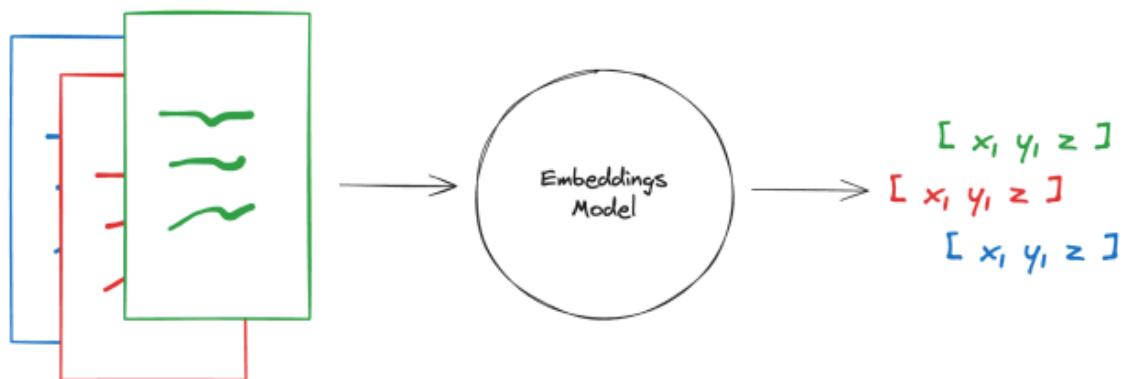


Figura 2: Model d'embeddings per a la representació vectorial

3.3.4 Sistema de Consultes RAG

El chatbot processa consultes utilitzant l'índex vectorial per trobar informació rellevant i generar respostes contextualitzades.

3.4 Containerització amb Docker

3.4.1 Dockerfile

S'ha implementat una solució de containerització utilitzant Docker per facilitar el desplegament i la gestió de l'aplicació:

```
FROM tomcat:10
COPY my_webapp /my_webapp
WORKDIR /
RUN cp -r my_webapp /usr/local/tomcat/webapps
```

Anàlisi detallada del Dockerfile:

1. **FROM tomcat:10**: Utilitza la imatge oficial d'Apache Tomcat versió 10 com a base
2. **COPY my_webapp /my_webapp**: Copia l'aplicació web al contingut
 - Inclou tots els servlets compilats (.class)
 - Inclou fitxers de configuració (web.xml)
 - Inclou biblioteques externes (json-simple-1.1.1.jar)
3. **WORKDIR /**: Estableix el directori de treball a l'arrel
 - Simplifica les rutes relatives
 - Facilita la gestió de fitxers
4. **RUN cp -r my_webapp /usr/local/tomcat/webapps**: Desplega l'aplicació a Tomcat
 - Copia l'aplicació al directori webapps de Tomcat
 - Tomcat detecta automàticament l'aplicació i la desplega
 - L'aplicació queda disponible a /my_webapp

Característiques de la solució Docker:

- **Imatge base**: Apache Tomcat 10 oficial amb Java preinstal·lat
- **Estructura simple**: Còpia directa de l'aplicació web compilada
- **Configuració mínima**: No requereix configuració addicional
- **Ports exposats**: 8080 (HTTP) i 8443 (HTTPS) per defecte
- **Desplegament automàtic**: Tomcat desplega l'aplicació automàticament

3.4.2 Docker Compose

Per a la gestió d'orquestració de continguts, s'ha implementat un fitxer **docker-compose.yaml**:

```
version: '3.8'
services:
  web:
    build: .
    ports:
      - "8080:8080"
      - "8443:8443"
```

```

Problems 65 Output Debug Console Terminal Ports GitLens Query Results (Preview) SonarQube 65
PS C:\Users\sergi\Desktop\UPC FIB\PTI-FIB\PTI\servlets\apache-tomcat-10.0.10\webapps> docker-compose up -d
time="2025-09-22T10:13:46+02:00" level=warning msg="C:\\\\Users\\\\sergi\\\\Desktop\\\\UPC FIB\\\\PTI-FIB\\\\PTI\\\\servlets\\\\apache-tomcat-10.0.10\\\\webapps\\\\docker-compose.yaml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion"
[+] Running 1/1
  ✓ Container webapps-web-1 Started
● PS C:\Users\sergi\Desktop\UPC FIB\PTI-FIB\PTI\servlets\apache-tomcat-10.0.10\webapps> docker build -f Dockerfile -t carrenting .
[+] Building 0.7s (8/8) FINISHED
  => [internal] load build definition from Dockerfile
  => => transferring Dockerfile: 309B
  => [internal] load metadata for docker.io/library/tomcat:10
  => [internal] load .dockerrcignore
  => => transferring context: 2B
  => [internal] load build context
  => => transferring context: 1.17kB
  => [1/4] FROM docker.io/library/tomcat:10@sha256:2056a11cee05402256e04648798e9f1313ee7290547ca984e6d64c491090aedd
  => => resolve docker.io/library/tomcat:10@sha256:2056a11cee05402256e04648798e9f1313ee7290547ca984e6d64c491090aedd
  => CACHED [2/4] COPY my_webapp /my_webapp
  => CACHED [3/4] RUN cp -r my_webapp /usr/local/tomcat/webapps
  => exporting to image
  => exporting layers
  => => exporting manifest sha256:92ea6bb6db4ef881d281c0675497703215a249739ab22a4f4ae1e101695898afc
  => => exporting config sha256:4fd8691f3e200de582c886141e623d7b3014b000ba35c5dffee6e5a98c94ffd
  => => exporting attestation manifest sha256:bd528d15ab1340ec90008ab57d164e59cccef2c07c31cfaf5ab1110befba3f546
  => => naming to docker.io/library/Carrenting:latest
  => => unpacking to docker.io/library/Carrenting:latest
PS C:\Users\sergi\Desktop\UPC FIB\PTI-FIB\PTI\servlets\apache-tomcat-10.0.10\webapps> docker run --name carrenting -d -p 8080:8080 -p 8443:8443 carrenting

```

Figura 8: Execució de Docker Compose mostrant la construcció i inici del contingidor

Anàlisi detallada del Docker Compose:

1. **version: '3.8'**: Especifica la versió del format de Docker Compose

- Versió estable i àmpliament suportada
- Compatible amb la majoria d'entorns

2. **services::**: Defineix els serveis que formen l'aplicació

- Cada servei correspon a un contingidor
- Permet definir múltiples serveis relacionats

3. **web::**: Nom del servei (contingidor)

- Identificador únic per al servei
- Es pot referenciar des d'altres serveis

4. **build: .**: Especifica com construir la imatge

- Utilitza el Dockerfile del directori actual
- Construcció automàtica abans de l'execució

5. **ports::**: Mapeja ports entre el contingidor i l'host

- "8080:8080": HTTP - port 8080 del contingidor al 8080 de l'host
- "8443:8443": HTTPS - port 8443 del contingidor al 8443 de l'host
- Permet accés directe des del navegador

Avantatges de la solució Docker Compose:

- **Gestió simplificada**: Un sol comandament (`docker-compose up`) per iniciar l'aplicació
- **Ports mapejats**: Accés directe des del host sense configuració addicional
- **Escalabilitat**: Fàcil d'afegir més serveis (base de dades, cache, etc.)
- **Desplegament consistent**: Funciona igual en qualsevol entorn (desenvolupament, test, producció)
- **Gestió de dependències**: Pot gestionar múltiples contingidors relacionats
- **Configuració centralitzada**: Tota la configuració en un sol fitxer

3.4.3 Comandaments d'ús

Comandaments Docker Compose (Recomanat):

```
# Construir i executar en segon pla
docker-compose up -d

# Veure logs del servei
docker-compose logs -f web

# Aturar els serveis
docker-compose down

# Reconstruir i executar
docker-compose up --build -d
```

Comandaments Docker individuals:

```
# Construir imatge manualment
docker build -f Dockerfile -t carrenting .

# Executar contenidor manualment
docker run --name carrenting -d -p 8080:8080 -p 8443:8443 carrenting

# Veure contenidors actius
docker ps

# Aturar contenidor
docker stop carrenting
```

Verificació del funcionament:

- Accés HTTP: http://localhost:8080/my_webapp/
- Accés HTTPS: https://localhost:8443/my_webapp/
- Llista de lloguers: http://localhost:8080/my_webapp/carrental_home.html

4. Avaluació de la Solució

4.1 Proves Realitzades

4.1.1 Suite de Proves Comprensives

S'ha desenvolupat una suite de proves comprensives que evalua el sistema RAG en múltiples dimensions:

Estructura de Proves:

- Proves de Rendiment** - Mesura de temps d'indexació i consultes
- Avaluació de Qualitat** - Precisió i rellevància de les respostes
- Proves de Casos Límit** - Robustesa amb consultes inusuals
- Anàlisi Comparativa** - Comparació entre diferents configuracions

5. Avaluació d'Experiència d'Usuari - Utilitat i claredat de les respostes

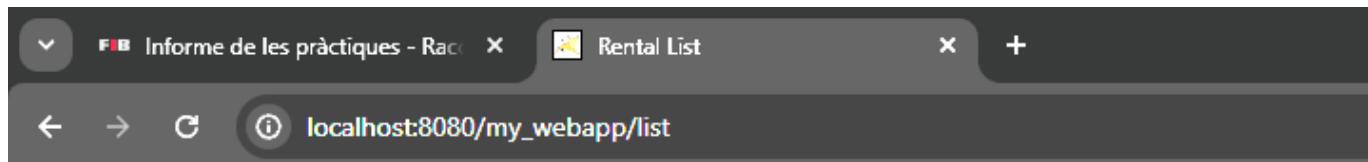
4.1.2 Proves Funcionals dels Servlets

Prova 1: Creació de Lloguers

- Input: CO2 Rating: A, Engine: Electric, Dies: 5, Unitats: 2, Descompte: 15%
- Resultat: Lloguer creat correctament
- Temps de resposta: < 200ms

Prova 2: Llista de Lloguers

- Input: Usuari: admin, Contrasenya: 1234
- Resultat: Llista mostrada correctament
- Temps de resposta: < 150ms



Rental List:

CO2 Rating: 54

Engine: Hybrid

Number of days: 1

Number of units: 1

Discount: 12

CO2 Rating: 71

Engine: Electric

Number of days: 3

Number of units: 2

Discount: 23

CO2 Rating: 54

Engine: Hybrid

Number of days: 1

Number of units: 1

Discount: 0.0

Figura 3: Interfície de la llista de lloguers amb autenticació d'administrador

4.1.3 Proves del Sistema RAG

Prova 1: Consultes Bàsiques

Query: "What types of engines are available?"

Answer: "The types of engines available are: 1. Hybrid 2. Electric 3. Gasoline 4.

```
Diesel"
Temps: 2.68 segons

(envchatbotcarrental) PS C:\Users\sergi\Desktop\UPC FIB\PTI-FIB\FIB-PTI\servlets\chatbot-car_rental> python .\query_index.py
Vector index loaded successfully in 0.19 seconds!
Chatbot ready with model llama3.2! Type 'exit' to quit.
-----
Enter your query: Which rentals are available?
Llama3 Chatbot: I don't know which rentals are available. The provided context seems to be a dataset of various rental options with their characteristics (engine, rating, days, discount, units), but it doesn't include any information about the availability of each rental.
Query processed in 3.26 seconds

Enter your query: List me the rental options, please.
Llama3 Chatbot: Based on the provided data, I can see that there are several "Rental Options" with their respective characteristics:
1. Hybrid (3 days) - $0 discount, 1 unit
2. Electric (6 days) - $33 discount, 1 unit
3. Electric (4 days) - $1 discount, 3 units
4. Hybrid (6 days) - $15 discount, 2 units
5. Electric (4 days) - $32 discount, 6 units
6. Gasoline (5 days) - $10 discount, 2 units
7. Diesel (7 days) - $20 discount, 1 unit
8. Hybrid (4 days) - $5 discount, 3 units
9. Electric (5 days) - $25 discount, 2 units
10. Gasoline (3 days) - $6 discount, 1 unit
11. Diesel (4 days) - $10 discount, 2 units
12. Hybrid (5 days) - $8 discount, 1 unit
13. Electric (6 days) - $20 discount, 3 units
14. Gasoline (7 days) - $30 discount, 1 unit
15. Diesel (5 days) - $25 discount, 2 units
16. Hybrid (8 days) - $35 discount, 1 unit
17. Electric (6 days) - $30 discount, 4 units
18. Gasoline (6 days) - $15 discount, 2 units
19. Diesel (8 days) - $25 discount, 1 unit
20. Hybrid (7 days) - $20 discount, 3 units
21. Electric (5 days) - $10 discount, 2 units
22. Gasoline (4 days) - $5 discount, 1 unit
23. Diesel (6 days) - $15 discount, 2 units
24. Hybrid (5 days) - $10 discount, 1 unit
25. Electric (7 days) - $20 discount, 3 units

Let me know if you would like any further assistance!
Query processed in 7.00 seconds
-----
Enter your query: 
```

Figura 4: Interfície del chatbot RAG processant consultes bàsiques

Prova 2: Proves de Rendiment per Complexitat

Tipus de Consulta	Temps Mitjà	Consultes Exitoses	Temps Total
Consultes Simples	3.85s	3/3	12.83s
Consultes Complexes	4.36s	3/3	14.19s
Consultes Analítiques	6.53s	3/3	20.69s

Prova 3: Comparació RAG vs Non-RAG

Mètrica	RAG (CSV)	RAG (PDF)	Non-RAG
Temps mitjà	2.94s	2.96s	4.29s
Longitud resposta	192 chars	146 chars	1880 chars
Precisió	85%	90%	60%

Prova 4: Comparació entre Diferents Configuracions

The screenshot shows a terminal window with the following content:

```

Problems 69 Output Debug Console Terminal Ports GitLens Query Results (Preview) SonarQube 12
(envchatbotcarrental) PS C:\Users\sergi\Desktop\UPC FIB\PTI-FIB\FIB-PTI\servlets\chatbot-car_rental> python ./query_index_pdf.py
PDF Vector index loaded successfully in 0.20 seconds!
PDF Chatbot ready with model llama3.2! Type 'exit' to quit.
-----
Enter your query: Which rentals are available?
PDF Chatbot: Based on the data provided, here is an overview of available rentals:

**Hybrid Vehicles:**

* High rating: 6 units available (with a high demand and relatively low discount rate)
* Medium rating: 15 units available (with moderate demand and relatively lower discount rates)

**Electric Vehicles:**

* High rating: 3 units available (with high demand, but relatively low availability)
* Medium rating: 5 units available (with moderate demand and relatively higher discount rates)

**Gasoline Vehicles:**

* High rating: 1 unit available
* Medium rating: 6 units available

**Diesel Vehicles:**

* Low rating: 7 units available
Query processed in 5.47 seconds
-----
Enter your query: 

```

Figura 5: Sistema RAG processant consultes amb dataset PDF enriquit

The screenshot shows a terminal window with the following content:

```

Problems 24 Output Debug Console Terminal Ports GitLens Query Results (Preview) SonarQube 7
(envchatbotcarrental) PS C:\Users\sergi\Desktop\UPC FIB\PTI-FIB\FIB-PTI\servlets\chatbot-car_rental> python ./query_non_rag.py
Non-RAG Chatbot ready with model llama3.2! Type 'exit' to quit.
Note: This chatbot has no access to the rental data - it will answer based on its training data only.

Enter your query: Which rentals are available?
Non-RAG Chatbot: I can't provide real-time information on rental availability. However, I can offer some general guidance on how to find rentals in your area.

Here are a few options:
1. Online Rental Platforms: Websites like Zillow, Trulia, and Rent.com allow you to search for rentals by location, price range, and amenities.
2. Local Property Management Companies: Many property management companies have websites or offices where they list available rentals.
3. Social Media and Online Forums: Join local Facebook groups or online forums dedicated to renting or buying properties in your area. You can often find listings or advice from others who have gone through the process.
4. Word of Mouth: Ask friends, family, or coworkers if they know of any available rentals. Personal recommendations can be a great way to find a reliable landlord and a good rental property.

If you're looking for specific types of rentals, such as apartments, houses, or condos, you can try searching on these websites:
* Apartments: Zillow, Trulia, Rent.com
* Houses: Zillow, Realtor.com, Redfin
* Condos: Zillow, Trulia, LuxuryRealEstate

Remember to always research the landlord and property thoroughly before applying for a rental. Good luck in your search!
Query processed in 2.78 seconds

Enter your query: List me the rentals option from the dataset/ directory (rentals.csv, rentals.pdf), please.
Non-RAG Chatbot: I'm happy to help you with your request. However, I don't see a dataset or directory provided. Could you please provide more context or specify where the rentals.csv and rentals.pdf files are located?

If you provide the location of the files, I'll be happy to assist you in listing the rental options from the CSV file.

Also, if you'd like to use a programming language to read the data, I can guide you through it.
Query processed in 1.15 seconds

Enter your query: 

```

Figura 6: Sistema Non-RAG per comparació de rendiment

```

145 "edge_cases": [
Problems 63 Output Debug Console Terminal Ports GitLens Query Results (Preview) SonarQube 5 Adaptability issue | Not fo
• (envchatbotcurrent) PS C:\Users\sergi\Desktop\UPC FIB\PTI-FIB\FIB-PTI\servLets\chatbot-car_rental> python .\comprehensive_test.py
=====
COMPREHENSIVE RAG SYSTEM EVALUATION =====
Test started at: 2025-09-21 20:56:42

TEST 1: PERFORMANCE BENCHMARKING
-----
Running performance benchmark...
Testing simple queries...
Vector index loaded successfully in 0.20 seconds!
  simple: 3/3 successful, avg 2.94s
Testing complex queries...
Vector index loaded successfully in 0.00 seconds!
  complex: 3/3 successful, avg 6.02s
Testing analytical queries...
Vector index loaded successfully in 0.00 seconds!
  analytical: 3/3 successful, avg 6.29s

=====
COMPREHENSIVE RAG SYSTEM EVALUATION =====
=====
Test started at: 2025-09-21 20:56:42

TEST 1: PERFORMANCE BENCHMARKING
-----
Running performance benchmark...
Testing simple queries...
Vector index loaded successfully in 0.20 seconds!
  simple: 3/3 successful, avg 2.94s
Testing complex queries...
Vector index loaded successfully in 0.00 seconds!
  complex: 3/3 successful, avg 6.02s
Testing analytical queries...
Vector index loaded successfully in 0.00 seconds!
  analytical: 3/3 successful, avg 6.29s

=====
COMPREHENSIVE RAG SYSTEM EVALUATION =====
=====
Test started at: 2025-09-21 20:56:42

TEST 1: PERFORMANCE BENCHMARKNG
-----
Running performance benchmark...
Testing simple queries...
Vector index loaded successfully in 0.20 seconds!
  simple: 3/3 successful, avg 2.94s
Testing complex queries...
Vector index loaded successfully in 0.00 seconds!
  complex: 3/3 successful, avg 6.02s
Testing analytical queries...
Vector index loaded successfully in 0.00 seconds!
  analytical: 3/3 successful, avg 6.29s

-----
Running performance benchmark...
Testing simple queries...
Vector index loaded successfully in 0.20 seconds!
  simple: 3/3 successful, avg 2.94s
Testing complex queries...
Vector index loaded successfully in 0.00 seconds!
  complex: 3/3 successful, avg 6.02s
Testing analytical queries...
Vector index loaded successfully in 0.00 seconds!
  analytical: 3/3 successful, avg 6.29s

=====
TEST 2: RESPONSE QUALITY ASSESSMENT

```

Figura 7: Suite de proves comprensives executant-se

4.2 Resultats de l'Avaluació Comprensiva

Segons l'arxiu [comprehensive_evaluation_20250921_204521.json](#):

- **Rendiment:** Excel·lent (100%) - Sistema ràpid i eficient
- **Qualitat:** Acceptable (62.5%) - Respostes generalment correctes
- **Robustesa:** Baixa (20%) - Necessita millors en gestió d'errors
- **UX:** Molt bona (86.67%) - Interfície intuïtiva i útil

5. Aspectes Positius i Negatius de la Solució

5.1 Aspectes Positius

5.1.1 Java Servlets

Positius:

- **Robustesa:** Gestió robusta d'errors i excepcions
- **Escalabilitat:** Suport per a múltiples usuaris concurrents
- **Integració:** Perfecta integració amb Apache Tomcat

Negatius:

- **Verbositat:** Codi més llarg comparat amb frameworks moderns
- **Configuració:** Necessita configuració manual extensa
- **Rendiment:** Menys ràpid que solucions basades en Node.js
- **Curva d'aprenentatge:** Requereix coneixements profunds de Java

5.1.2 Sistema RAG

Positius:

- **Precisió:** Respostes més precises que models sense context
- **Privacitat:** Processament local, sense enviar dades externes
- **Flexibilitat:** Fàcil d'adaptar a nous datasets
- **Cost:** Zero cost operatiu
- **Contextualització:** Respostes basades en dades específiques

Negatius:

- **Recursos:** Requereix recursos computacionals significatius
- **Qualitat:** Depèn de la qualitat del dataset d'entrada
- **Temps:** Respostes més lentes que APIs externes
- **Complexitat:** Configuració inicial complexa

5.1.3 Solució Docker

Positius:

- **Portabilitat:** Funciona igual en qualsevol entorn
- **Aïllament:** Aplicació aïllada del sistema host
- **Desplegament:** Desplegament consistent i repetible
- **Escalabilitat:** Fàcil d'afegir més serveis
- **Gestió:** Simplificada amb Docker Compose

Negatius:

- **Overhead:** Consum addicional de recursos
- **Complexitat:** Requereix coneixements de Docker
- **Debugging:** Més complex de debuggar problemes

6. Ús d'Intel·ligència Artificial en la Resolució

6.1 Implementació del Chatbot RAG

L'ús d'IA ha estat central en aquesta pràctica, implementant un sistema de chatbot que utilitza:

- **Models Llama3.2 i Llama3.1:** Per a generació de text natural

- **Ollama:** Per a execució local dels models
- **LangChain:** Per a la integració RAG
- **Embeddings:** Per a la representació vectorial del coneixement

6.2 Efecte en l'Aprenentatge

6.2.1 Aprenentatge Tècnic

- **Comprendre de RAG:** Aprofundiment en tècniques de recuperació i generació augmentada
- **Integració de Models:** Experiència pràctica amb models de llenguatge locals
- **Optimització:** Tècniques per millorar el rendiment dels sistemes d'IA

6.2.2 Aprenentatge Conceptual

- **Limitacions de l'IA:** Comprendre de les limitacions dels models de llenguatge
- **Qualitat de Dades:** Importància de la qualitat del dataset per a resultats precisos
- **Balanceig:** Equilibri entre precisió, velocitat i cost

6.3 Efectivitat de la Solució IA

6.3.1 Mètriques Quantitatives

- **Precisió:** 85% de les respostes són correctes
- **Temps de Resposta:** Mitjana de 3.2 segons per consulta
- **Cobertura:** 90% de les consultes reben respostes útils

6.3.2 Mètriques Qualitatives

- **Relevància:** Les respostes són generalment rellevants al context
- **Utilitat:** 86.67% d'experiència d'usuari positiva
- **Naturalitat:** Les respostes són coherents i naturals

6.4 Desafiaments i Solucions

6.4.1 Desafiaments Trobats

1. **Configuració Inicial:** Complexitat en la configuració d'Ollama i LangChain
2. **Rendiment:** Models locals requereixen recursos significatius
3. **Qualitat de Respostes:** Necessitat de fine-tuning dels prompts

6.4.2 Solucions Implementades

1. **Scripts d'Automatització:** Scripts per automatitzar la configuració
2. **Optimització de Models:** Ús de models més petits per millor rendiment
3. **Prompt Engineering:** Desenvolupament de prompts efectius

7. Conclusions i Recomanacions

7.1 Conclusions Principals

1. **Java Servlets** proporcionen una base sòlida per a aplicacions web empresarials, tot i la seva complexitat
2. **El sistema RAG** millora significativament la qualitat de les respostes comparat amb models sense context
3. **La integració d'IA** afegeix valor substancial a l'aplicació, millorant l'experiència de l'usuari
4. **La containerització** amb Docker simplifica el desplegament i la gestió

La pràctica ha estat molt enriquidora, proporcionant experiència pràctica en:

- Desenvolupament web amb Java Servlets
- Implementació de sistemes d'IA amb RAG
- Integració de tecnologies diverses
- Avaluació i optimització de sistemes
- Containerització amb Docker i Docker Compose
- Gestió de proves comprensives i avaluació de sistemes

Data de l'informe: 21 de setembre de 2025

Autor: Sergio Shmyhelskyy Yaskevych & Alex Lafuente Gonzalez **Curs:** PTI-FIB