

AIFA scraper backend

Software Architectures and Methodologies

Federico Magnolfi

Ingegneria Informatica
Università degli studi di Firenze

March 2021





Table of Contents

1. Introduction
2. AIFA services
3. Software architecture
4. Software usage
5. Conclusions



Introduction

Agenzia Italiana del Farmaco

- public institution
- responsible for the regulatory activity of pharmaceuticals in Italy

Medicines data

AIFA maintains the only official database of Italian medicines, which contains information like names, companies, active ingredients, leaflets...

This project

How can we query AIFA database?

There is a service, though it is not well-documented

Objective of this project

- **extract** data from AIFA services
- **store** data in a relational database
- **provide** access to data through a REST API

Technologies

- JEE with Hibernate
- MariaDB
- Gradle and Docker



AIFANZ services



AIFA web interface

The screenshot shows a search results page for the term "betadine". The page has a dark blue header with navigation links: HOME, FARMACO, PRINCIPIO ATTIVO, AZIENDA, and AGGIORNAMENTI. Below the header are three search icons: "Cerca Farmaco" (F), "Cerca Princípio Attivo" (PA), and "Cerca Azienda" (A). A search bar contains the query "betadine". The results are listed in two columns:

Nome	AIC	Nome	AIC
BETADINE	023907	BETADINE	037114
Azienda: MEDA PHARMA S.P.A.		Azienda: FARMA 1000 S.R.L.	
BETADINE	037960	BETADINE	037981
Azienda: BB FARMA S.R.L.		Azienda: PROGRAMMI SANITARI INTEGRATI S.R.L.	
BETADINE	038397	BETADINE	039174
Azienda: PHARMAZENA S.R.L.		Azienda: LINK PHARM S.R.L.	
BETADINE	040978	BETADINE	041676
Azienda: GEKOFAR S.R.L.		Azienda: BEACHCOURSE ITALIA S.R.L.	
BETADINE	041723	BETADINE	042073
Azienda: MPF PHARMA S.R.L.		Azienda: 4PHARMA S.R.L. IN BREVE DETTA ANCHE FP S.R.L.	
BETADINE	042858	BETADINE	042901
Azienda: GMM FARMA S.R.L.		Azienda: NEW PHARMASHOP S.R.L.	
BETADINE	042991	BETADINE	044140
Azienda: FARMAVOX S.R.L.		Azienda: FARMED S.R.L.	
BETADINE	045462	BETADINE	045323
Azienda: FARMAROC S.R.L.		Azienda: GLOBAL PHARMACIES PARTNER HEALTH S.R.L.	

At the bottom of the results page are navigation buttons: <<, <, 1, 2, >, >>.

Agenzia Italiana del Farmaco - Via del Tritone, 181 - 00187 Roma - tel. 06 5978401

By inspection:

- one remote endpoint
- example queries
- response fields
- wildcard *

On parameters:

- many response formats
- can specify fields
- can limit the results



Understanding of AIFA services

GET https://www.agenziafarmaco.gov.it/services/search/select?q=2021+bundle:confezione_farmaco&wt=csv&start=130&rows=10&fl=sm_field_codice_farmaco,sm_field_stato_farmaco,sm_field_descrizione_confezione,sm_field_link_fi

Send

Params • Authorization Headers (6) Body Pre-request Script Tests • Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/> q	2021+bundle:confezione_farmaco	query		
<input checked="" type="checkbox"/> wt	csv	response format		
<input checked="" type="checkbox"/> start	130	number of results to skip		
<input checked="" type="checkbox"/> rows	10	number of returner results		
<input checked="" type="checkbox"/> fl	sm_field_codice_farmaco,sm_field_stato_farmaco,sm_field_descrizione_confezione,sm_field_link_fi	list of returned fields		
Key	Value	Description		

Body Cookies Headers (4) Test Results

200 OK 661 ms 4.85 KB Save Response

Pretty Raw Preview Visualize

sm_field_codice_farmaco	sm_field_stato_farmaco	sm_field_descrizione_confezione	sm_field_link_fi
046453	A	"300 MG COMPRESSE" 60 COMPRESSE IN BLISTER PVC-AL	https://farmaci.agenziafarmaco.gov.it/aifa/pdfFileName=footer_004852_046453_Fi.pdf
046453	A	"300 MG COMPRESSE" 100 COMPRESSE IN BLISTER PVC-AL	https://farmaci.agenziafarmaco.gov.it/aifa/pdfFileName=footer_004852_046453_Fi.pdf

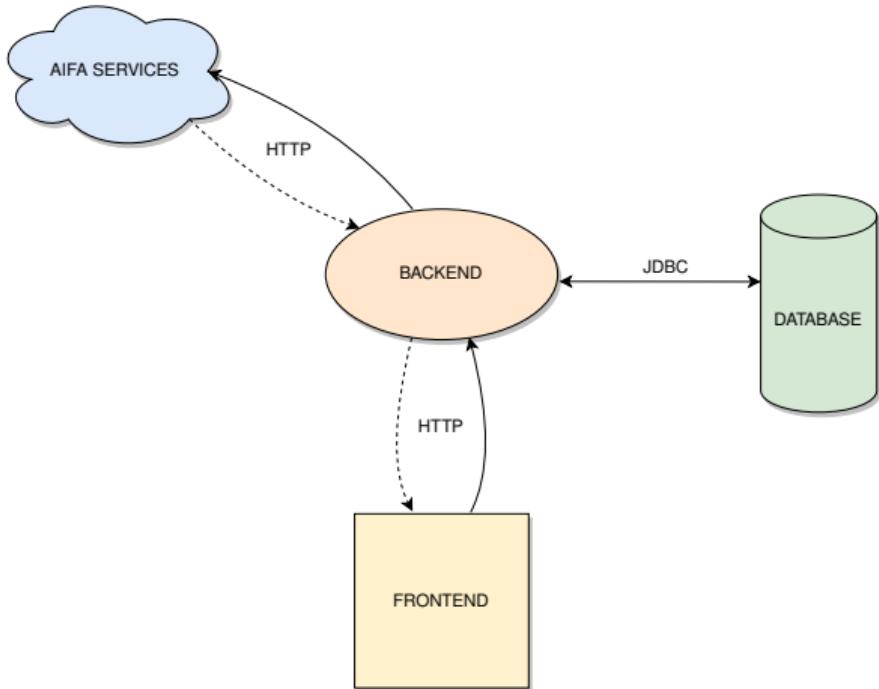
Discoveries:

- other parameters
- AND/OR expressions
- paginated requests
- filter by year



Software architecture

System architecture



What already exists:

- AIFA services
- frontend

What we developed:

- backend
- database

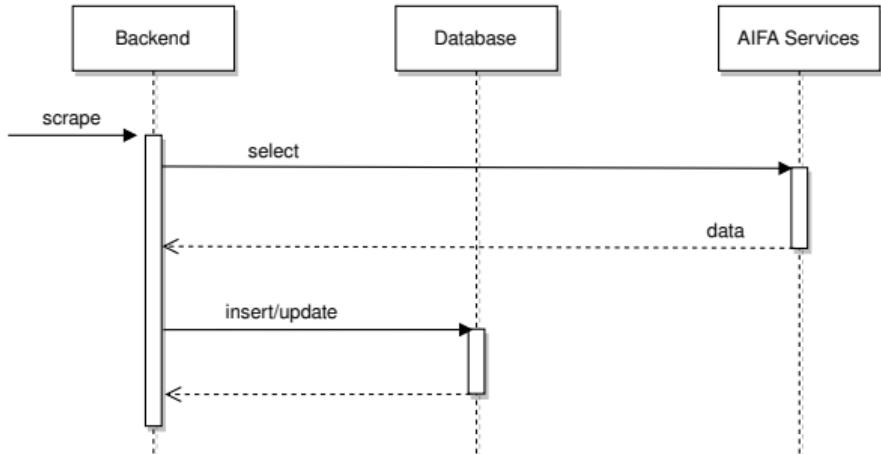
Interactions during scrape

Scrape procedure:

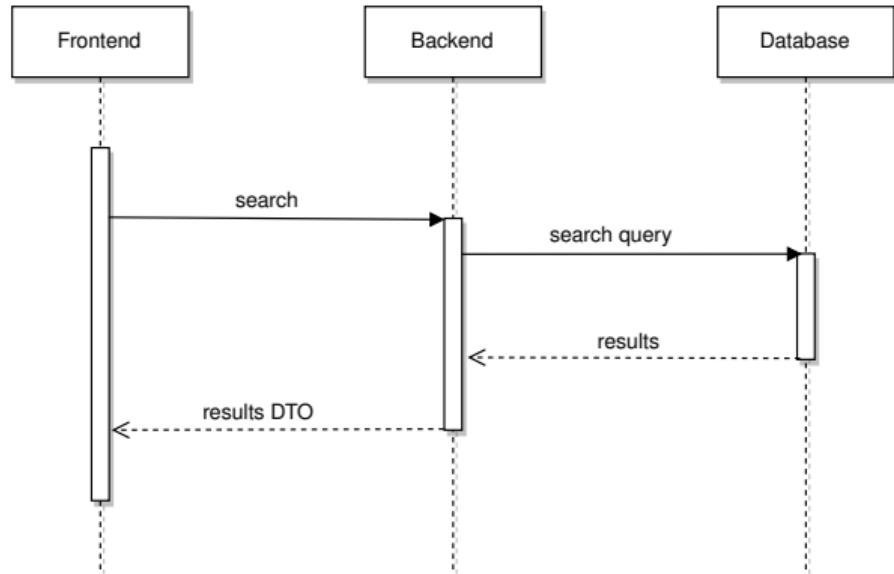
1. get data from aifa services
2. elaborate data
3. save data into the database

Suggested start automation:

- manual start: using cURL
- periodic start: using cron



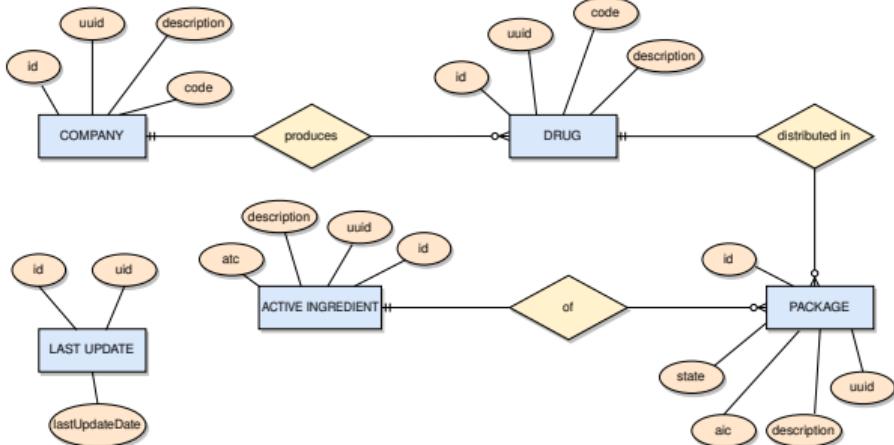
Interactions during search



Search steps

1. frontend send the request
2. backend queries the database
3. results are returned in JSON

Entity-Relationship diagram



Entities that store data:

- Active Ingredient
- Company
- Drug
- Package

Last Update entity to reduce load

Remark

Active Ingredient is linked to package

Note

Tables are auto-generated with JPA

Server load during scrape

Several techniques are available to reduce the load of the server

Reduce downloaded data

Download only useful fields from AIFA

Paginate downloads

Download chunk after the preceding was processed

Paginate transactions

Many insert in the same transaction

Select year

Download only records modified in the year

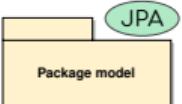
Discard old

Discard records not modified since last scrape

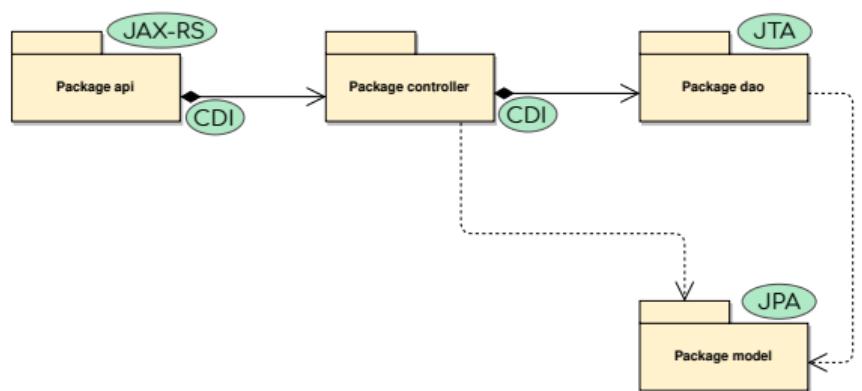


Packages

- **model:** domain model
- **api:** exposes the endpoints
- **controller:** business logic
- **dao:** provides data operations
- **requester:** handle requests to AIFA services
- **dto:** defines the format of JSON objects
- **mapper:** converts domain objects to DTOs
- **config:** enables CORS

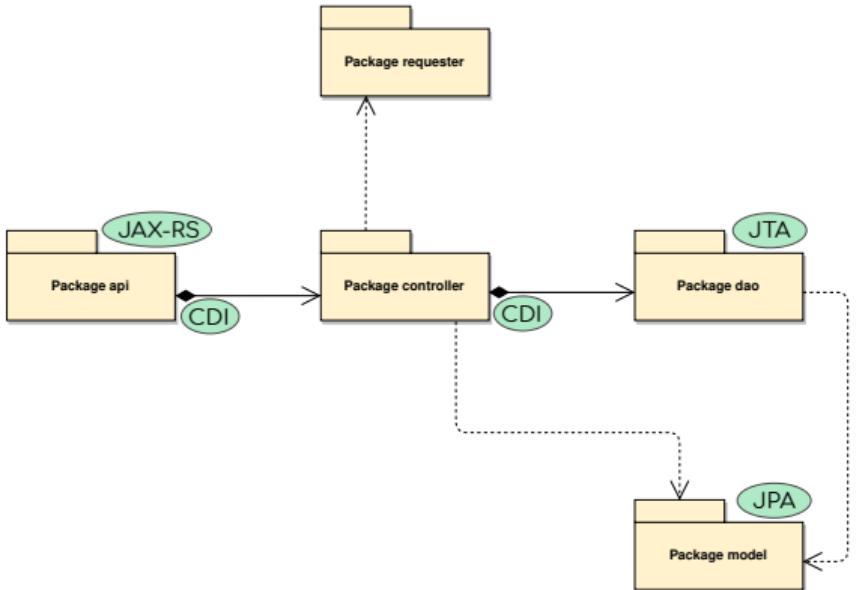


Packages



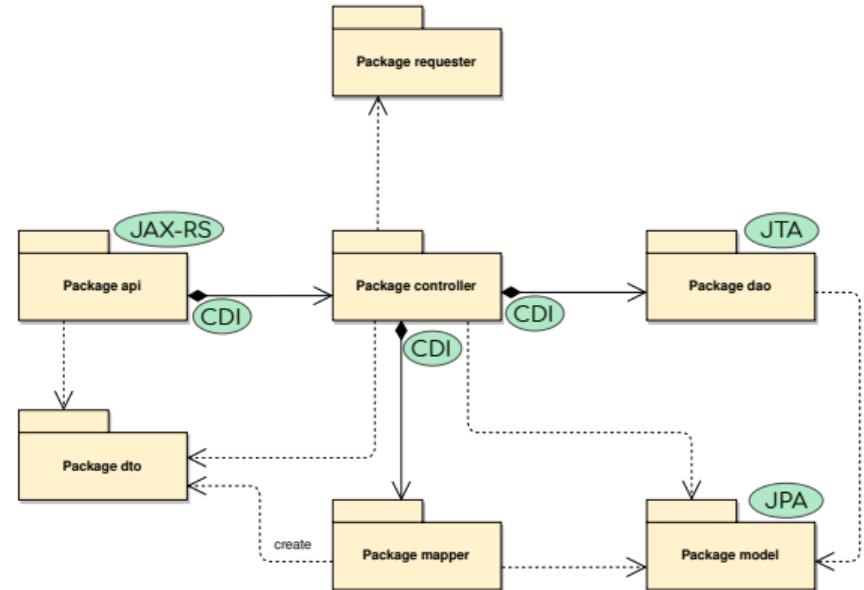
- **model:** domain model
- **api:** exposes the endpoints
- **controller:** business logic
- **dao:** provides data operations
- **requester:** handle requests to AIFA services
- **dto:** defines the format of JSON objects
- **mapper:** converts domain objects to DTOs
- **config:** enables CORS

Packages



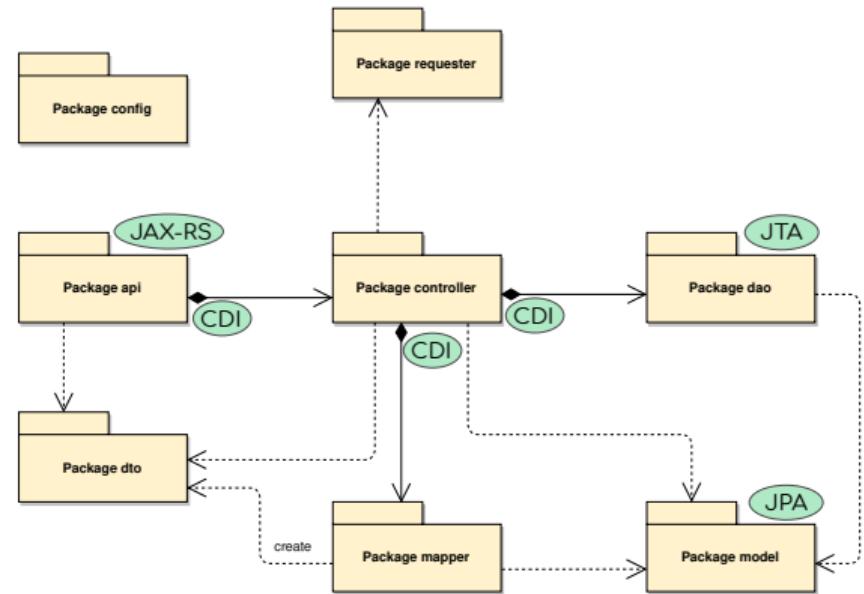
- model: domain model
- api: exposes the endpoints
- controller: business logic
- dao: provides data operations
- **requester:** handle requests to AIFA services
- dto: defines the format of JSON objects
- mapper: converts domain objects to DTOs
- config: enables CORS

Packages



- **model: domain model**
- **api: exposes the endpoints**
- **controller: business logic**
- **dao: provides data operations**
- **requester: handle requests to AIFA services**
- **dto: defines the format of JSON objects**
- **mapper: converts domain objects to DTOs**
- **config: enables CORS**

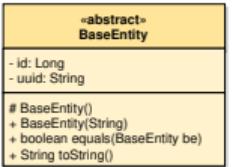
Packages



- **model**: domain model
- **api**: exposes the endpoints
- **controller**: business logic
- **dao**: provides data operations
- **requester**: handle requests to AIFA services
- **dto**: defines the format of JSON objects
- **mapper**: converts domain objects to DTOs
- **config**: enables CORS

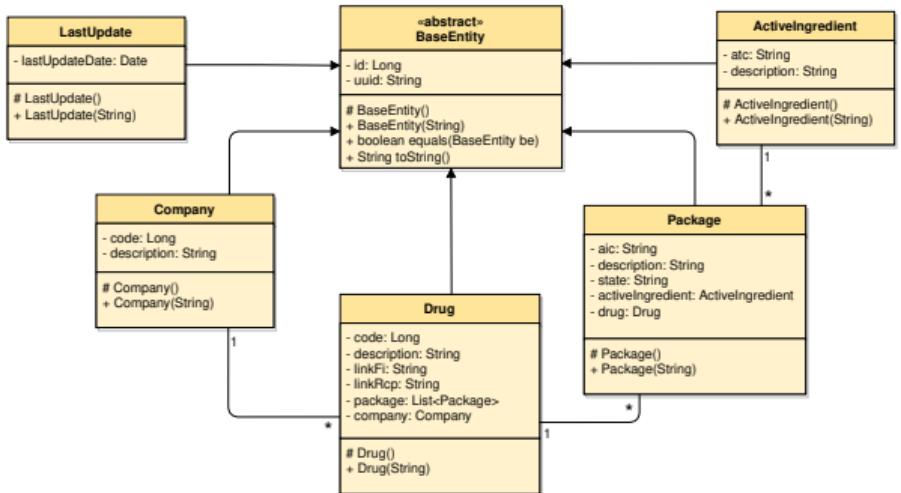


Domain model



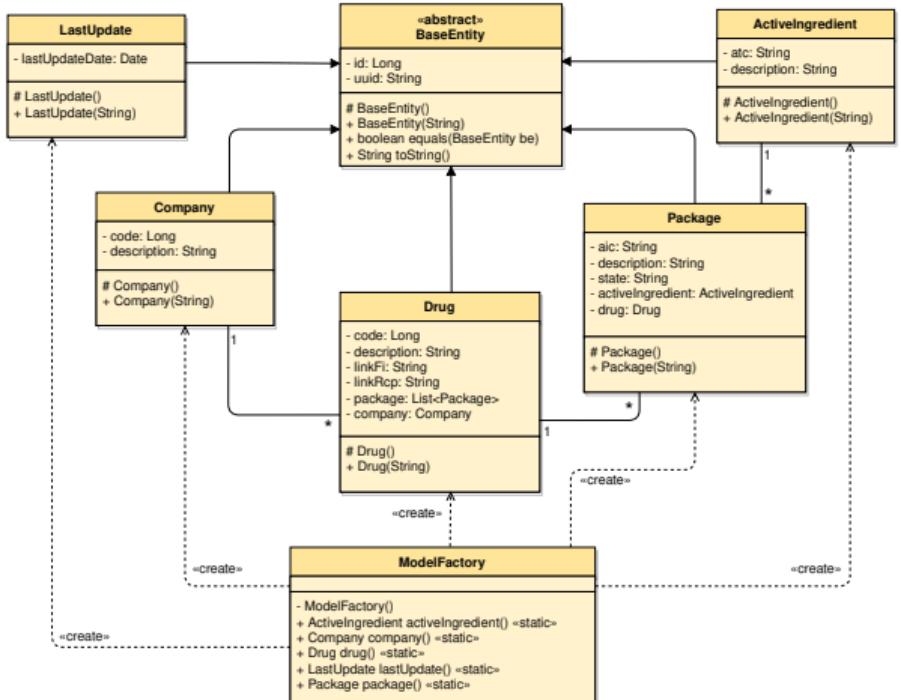
- a common abstract base class

Domain model



- a common abstract base class

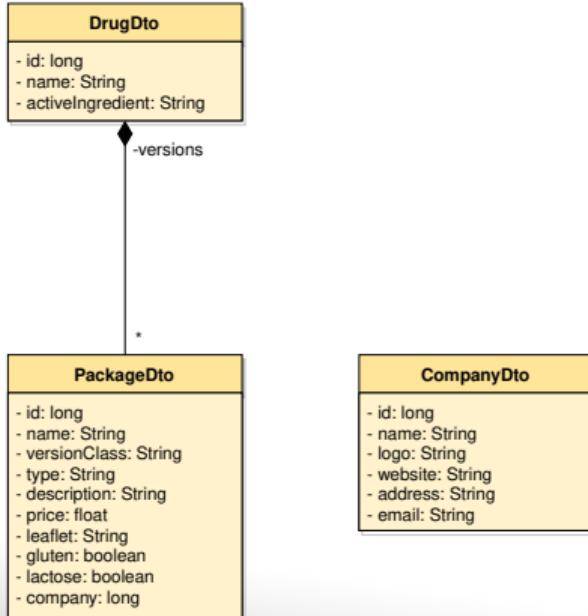
Domain model



- a common abstract base class
- a factory creates all the entities

getters and setters omitted

Data Transfer Objects



Compatibility

The client expect objects in different format

Solution

Use the Data Transfer Object pattern

1. a mapper convert an entity into a DTO
2. DTOs are transformed into JSON

Scrape API

PUT {server_url}/api/scrape

Parameter	Type	Default	Description
<i>url</i>	String	null*	url of remote service
<i>file</i>	String	null	path to csv file
<i>firstResult</i>	Integer	0	index of first result in download
<i>maxResults</i>	Integer	10	max results in all downloads
<i>downloadSize</i>	Integer	<i>maxResults</i>	max results per download
<i>transactionSize</i>	Integer	1000	max insert per transaction
<i>checkLastUpdate</i>	boolean	false	if true, discard records not updated since last time
<i>year</i>	Integer	null	request records with year in one of the date fields



Search API

GET {server_url}/api/search

Parameter	Type	Default	Description
<i>drug</i>	String	null	value to search in drug description
<i>name</i>	String	null	alias for drug (compatibility reasons)
<i>activeIngredient</i>	String	null	value to search in active ingredient description
<i>company</i>	String	null	value to search in company description
<i>text</i>	String	null	value to search in all descriptions
<i>firstResult</i>	Integer	0	index of first result
<i>maxResults</i>	Integer	null	number of maximum results
<i>retired</i>	boolean	false	if true, return retired packagings too

Type of returned JSON: List<DrugDto>





Software usage

Persistence unit configuration: persistence.xml

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<persistence xmlns="http://xmlns.jcp.org/xml/ns/persistence"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
        http://xmlns.jcp.org/xml/ns/persistence/persistence_2_2.xsd" version="2.2">
    <persistence-unit name="aifaPU" transaction-type="JTA">
        <jta-data-source>java:jdbc/datasources/AifaDB</jta-data-source>
        <properties>
            <property name="hibernate.dialect" value="org.hibernate.dialect.MariaDBDialect" />
            <property name="hibernate.hbm2ddl.auto" value="update"/>
            <property name="hibernate.show_sql" value="true"/>
        </properties>
    </persistence-unit>
</persistence>
```

Remark

The persistence unit configuration is Hibernate-specific.



Datasource configuration: web.xml

```
<data-source>
    <name>java:jdbc/datasources/AifaDB</name>
    <class-name>org.mariadb.jdbc.MariaDbDataSource</class-name>
    <server-name>aifadb-service</server-name>
    <port-number>3306</port-number>
    <database-name>aifa_db</database-name>
    <user>root</user>
    <password></password>
</data-source>
```

Remark

The `name` property shouldn't be changed: it is linked to `persistence.xml`.
The other properties can be changed, according to the desired datasource.

Build

Gradle

- no installation needed (thanks to the gradle wrapper)
- fast and easy



Compile the application

- Linux/macOS: `./gradlew war`
- Windows: `gradle.bat war`

Compilation output

`aifa-scraper.war` inside `build/libs/` subdirectory

Run: two different ways

Without docker

1. start the DBMS
2. create a database
3. modify datasource configuration in *web.xml*
4. compile the application with *Gradle*
5. run *WildFly* in standalone mode
6. move *aifa-scraper.war* into *standalone/deployments/* subdir of *WildFly*

With docker

1. compile the application with *Gradle*
2. launch with *Docker Compose*: `docker-compose up`



Backend

In the *Dockerfile*:

1. start from a *WildFly* image
2. move the `aifa-scraping.war` file
3. run *WildFly* in standalone mode

All together

The `docker-compose.yml` defines:

1. a service based on the built backend
2. a service based on a MariaDB image
3. a network that allows the two services to communicate with each other



Conclusions



Recap

- JEE server
- scrape data from AIFA
- control server load during scrape
- serve data through a RESTful API
- easy compilation and running with Gradle and Docker



Thanks for the attention!

Dockerfile

```
FROM jboss/wildfly:22.0.1.Final

ADD build/libs/aifa-scraper.war /opt/jboss/wildfly/standalone/deployments/

CMD ["/opt/jboss/wildfly/bin/standalone.sh", "-b", "0.0.0.0", "-bmanagement", "0.0.0.0"]

EXPOSE 8080 9990 5005
```





docker-compose.yml

```
version: "3.7"
services:
  jeeserver-service:
    container_name: "aifa-scraper_jeeserver_container"
    build:
      context: .
    depends_on:
      - aifadb-service
    networks:
      - aifa-scraper-network
    ports:
      - "8080:8080" # application
      - "9990:9990" # admin console
      - "5005:5005" # debug port
  aifadb-service:
    container_name: "aifa-scraper_db_container"
    image: "mariadb:10.5.8"
    environment:
      - MYSQL_DATABASE=aifa_db
      - MYSQL_ALLOW_EMPTY_PASSWORD=true
    networks:
      - aifa-scraper-network
    ports:
      - "3306:3306"
networks:
  aifa-scraper-network:
    name: aifa-scraper-network
```