



Full-Text Search on localized items

Hibernate Search for full-text search on
localization-related ORM entities

Software Architectures and Metodologies
Prof. Enrico Vicario

Francesco Areoluci

Project Overview

Project aim

The aim of the project is to address and investigate about the following problems:

1. Develop a **RESTful backend application** based on Java EE (JPA + CDI) to perform full-text search using the Hibernate Search library...
2. ...on a logic domain that implements **internationalization/localization** of entities. Hibernate Search should manage indexing and searching of the localized Object Relational Mapped entities.



Project Overview

Internationalization and Localization [1]

- **Localization**, often referred as **l10n** where 10 is the number of letters between *l* and *n*, is the process of **adapting** a product, an application or a textual document to a specific country or region.
- The process of localization of a product or application can be easily enabled through the process of internationalization.
- **Internationalization**, often referred as **i18n** where 18 is the number of letters between *i* and *n*, is the process of design and development of a product or application that enables **easy localization**.



Project Overview

Hibernate Search [2]

- Enabling technology for **indexing** and **full-text search** on Hibernate applications. It allows to extract data from hibernate ORM entities and push it to local or remote indexes.
- Hibernate Search is particularly useful for applications where SQL-based searches are not suited, such as full-text and geolocation searches.
- The search is based on the concept of **inverted indexes**: a dictionary where the key is a token found in a document and the value is the list of identifiers of every document containing that token



Project Overview

Hibernate Search

- Hibernate Search backend module abstracts the full-text search engine, by implementing indexing and searching interfaces.
- Provides abstractions based on two popular full-text search libraries: **Lucene** (local indexing) and **Elasticsearch** (remote indexing).

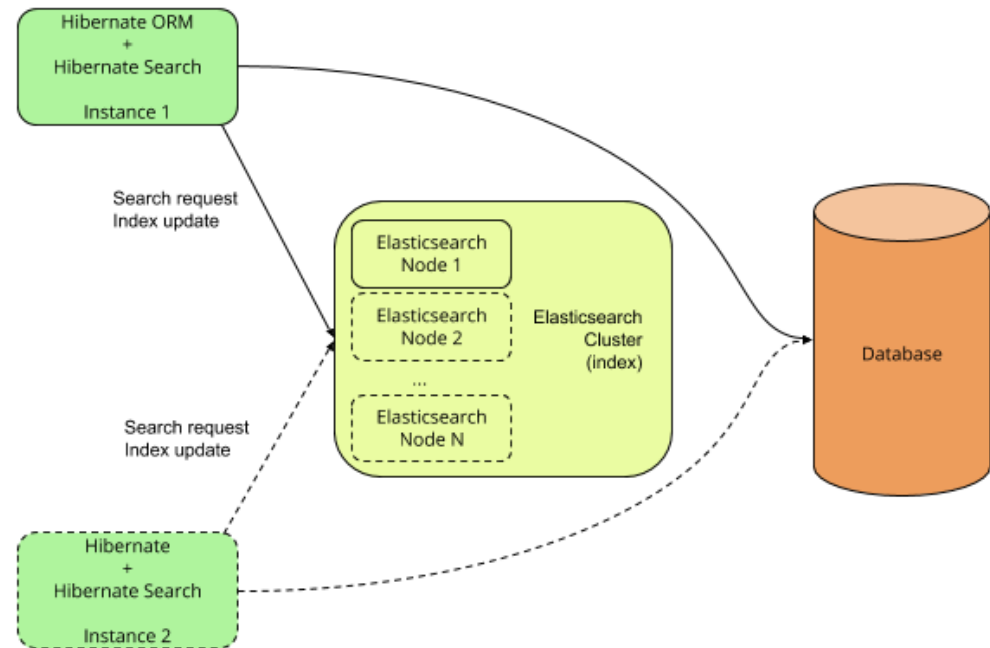


Figure 1: Hibernate Search's Elasticsearch backend

Context of application

Full-text search on localized entities

- The developed application represents the backend of an online shop which let the users to **purchase products** and **search** for them through query on their name and/or description.
- **Products fields**, such as name and description, must be **localized** in multiple locales: English and Italian.
- Through this application, Hibernate Search functionalities can be used on product's localized attributes: these fields in multiple languages will be used by Hibernate Search to retrieve matching entities.
- The system should offer its functionalities through a **RESTful interface** using the **JAX-RS** standard.

Workflow

Steps

The project objectives have been accomplished through the following workflow:

1. Application requirements definition;
2. Application use cases design;
3. Application class design;
4. Endpoints definition;
5. Sequence diagrams design;
6. Application development and testing



Workflow

Requirements definition

Requirements have been designed in order to define what operations the system should do and how these operations should be accomplished. To do that, four types of requirements have been used:

- Functional Requirements;
- Non-functional requirements;
- Domain requirements;
- Project constraints



Workflow

Requirements definition

- **FR1:** The system must allow to manage the available products
- **FR2:** Each product must contains the following fields:
 - **FR2.1:** Product Name
 - **FR2.2:** Product Description
 - **FR2.3:** Price
 - **FR2.4:** Manufacturer
 - **FR2.5:** Product category
- **FR3:** The system must have the localization feature: products must be localized in multiple languages (it, en). Localization feature must be used on the following fields:
 - **FR3.1:** Product Name
 - **FR3.2:** Product Description
 - **FR3.3:** Price
 - **FR3.4:** Product Category
- **FR4:** The system must allow the visualization and purchase of the products.
 - **FR4.1:** The product visualization must be localized: a product must be returned to the user with fields specified in FR3 properly localized.
- **FR5:** The system must features the search on products.
- **FR6:** The system must features the management of two types of user account
 - **FR6.1:** Administrator account - Responsibilities: Product management and research, as specified in FR1 and FR5
 - **FR6.2:** Customer account - Responsibilities: Product research and purchase, as specified in FR4 and FR5

Figure 2: Application Functional Requirements

Workflow

Requirements definition

- **NFR1** (Security Requirement): Application access must be managed through user authentication for all the account types described in FR6.
- **NFR2** (Implementation Requirement): The purchase functionality, as specified in FR4, must be prototyped as following:
 - **NFR2.1**: The user must have a shopping cart. The user can add and remove products to/from the cart;
 - **NFR2.2**: Once the user has added to the cart the desired products, he/she can proceed to the checkout. This operation will move the cart products to a shopping list. The cart will be cleared.
- **DR1**: The product management, as specified in FR1, must allow to add, remove and edit the products (CRUD operations)
- **DR2**: The products search, as specified in FR5, must be based on query keywords and must be used to search on products' name and description.
- **DR3**: A customer account must have an associated locale in order to implement the product visualization feature.

Figure 3: Application Non-Functional Requirements

Figure 4: Application Domain Requirements

Workflow

Requirements definition

- C1: The application must be developed using Java EE, with JPA and CDI technologies.
- C2: The Persistence layer must be managed using Hibernate.
- C3: The persistence must be managed by DBMS MariaDB.
- C4: The search layer must be manager through Hibernate Search library.
- C5: Application services must be exposed via REST API using JAX-RS technology.
- C6: Package management must be managed through Maven.

Figure 5: Project constraints



Workflow

Use case design

- Use case definition has allowed to formalize the actions that each actor (customer or administrator) is allowed to do within the application.
- Each action can include or be extended by other actions.

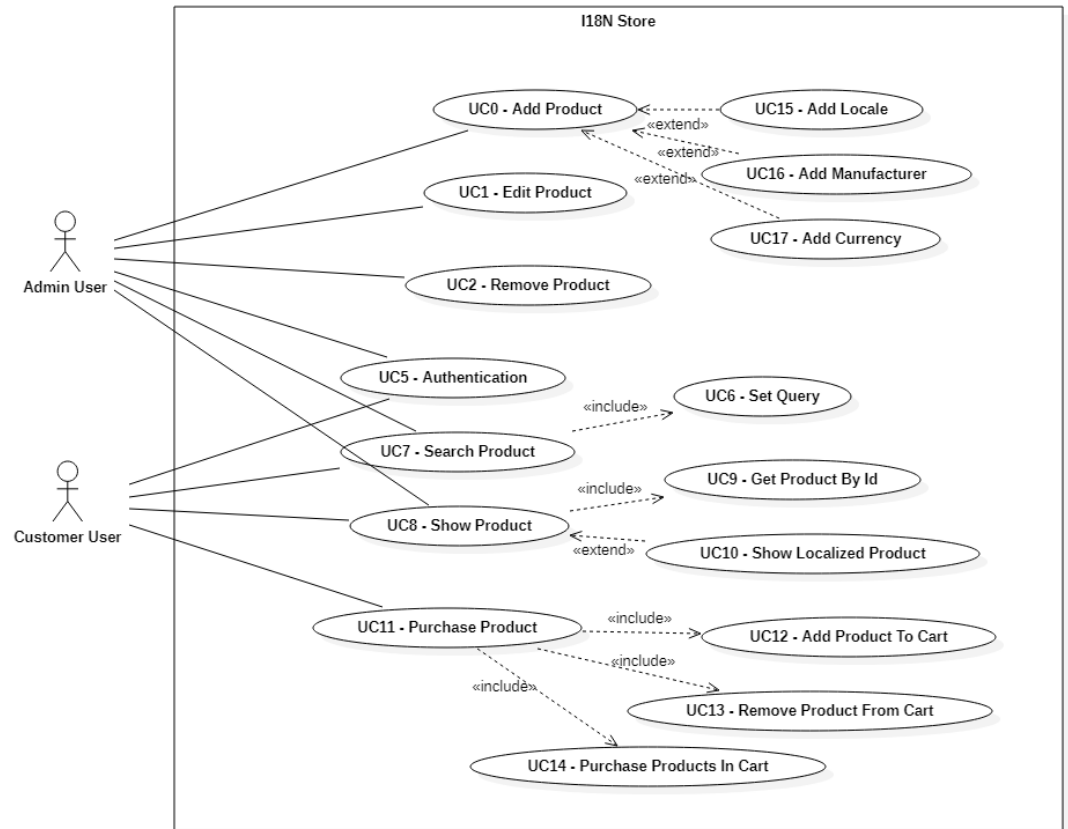


Figure 6: Application use cases

Workflow

Class design

The application has been modeled through the UML class diagram. The application has been divided in packages, one for each logic domain:

- **Domain Model:** entities that represents the application domain;
- **Translation Model:** entities responsible of the field translation handling;
- **Rest:** contains the available controllers (endpoints);
- **Data Access Objects:** entities responsible for DBMS interaction and instantiation of domain model entities;
- **Data Transfer Objects:** entities used to transfer data from endpoints to client and viceversa;
- **Security:** entities responsible of user authentication and authorization management.





Workflow

Domain Model Package

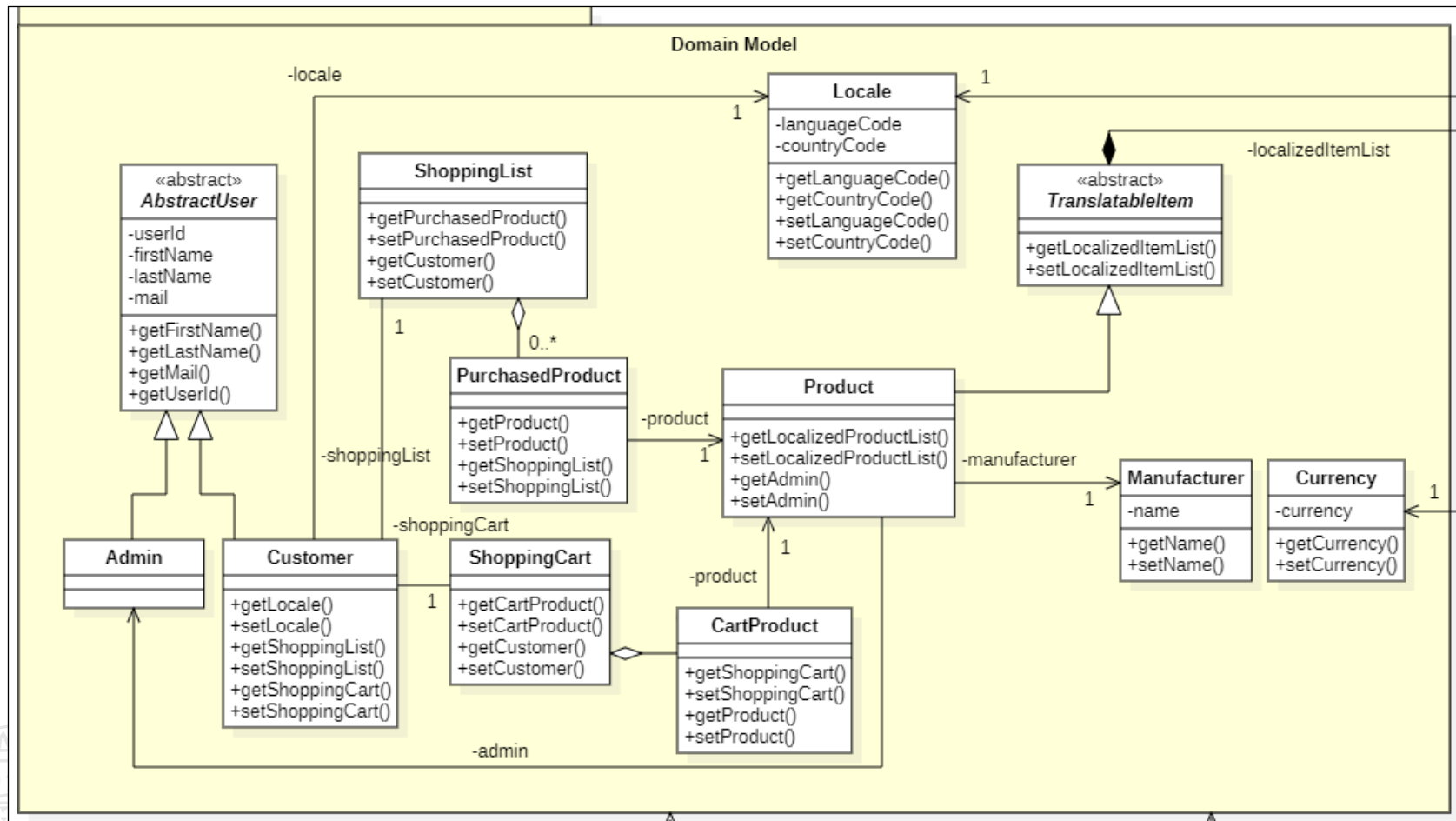


Figure 7: Domain Model package class diagram

Workflow

Translation Model Package

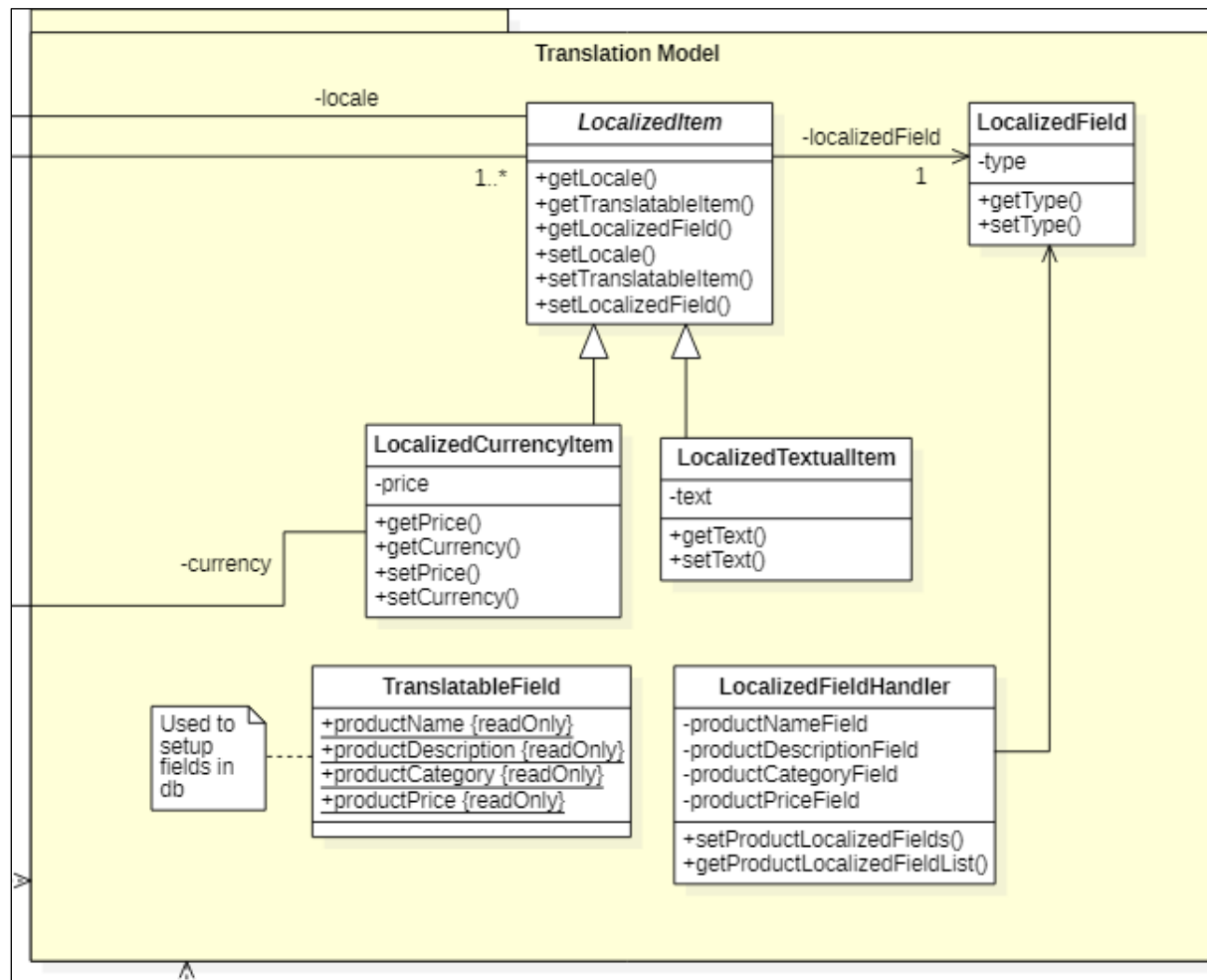


Figure 8: Translation Model package class diagram

Workflow

DAO Package

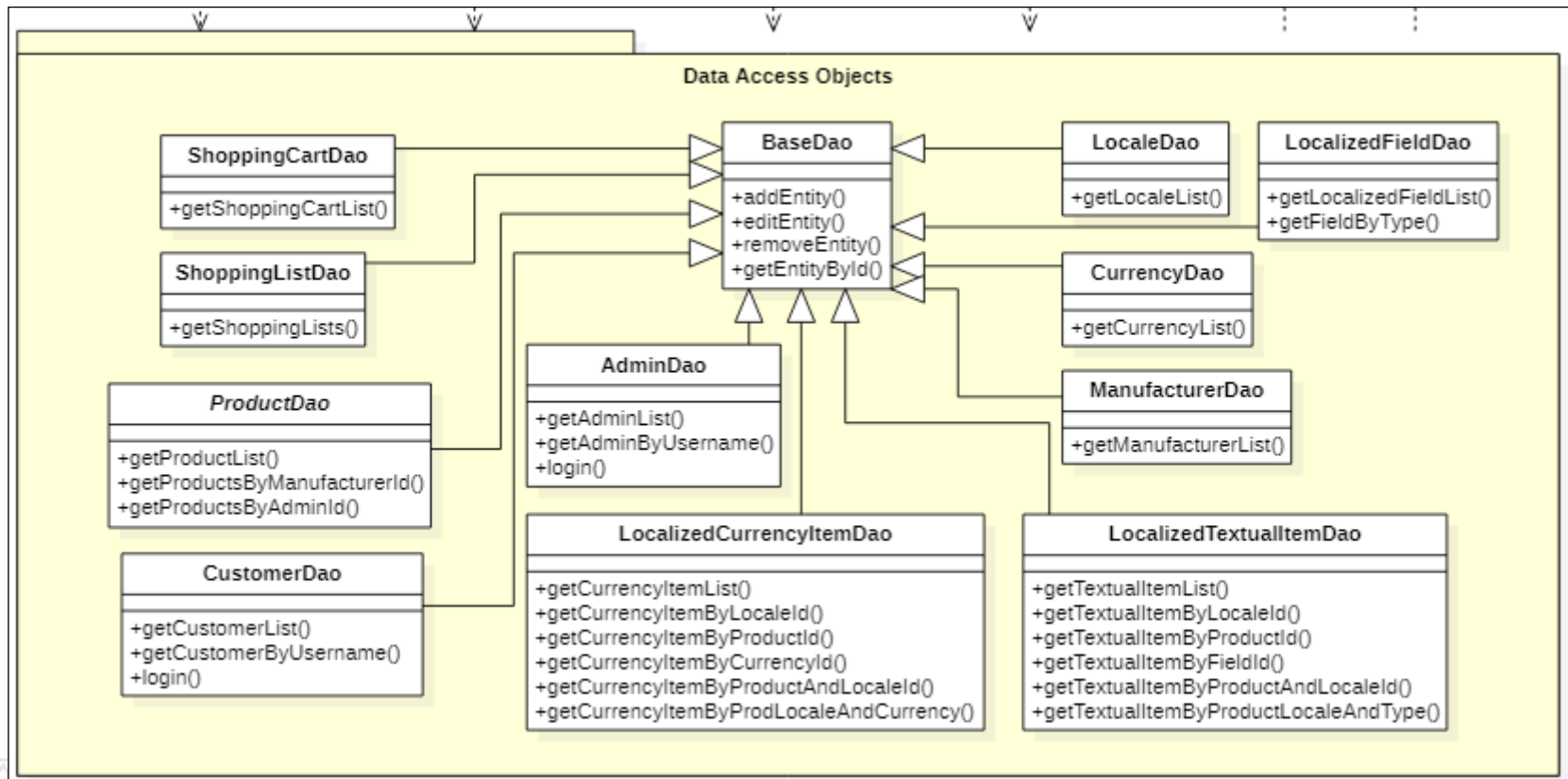


Figure 9: Data Access Object package class diagram

Workflow

Rest Package

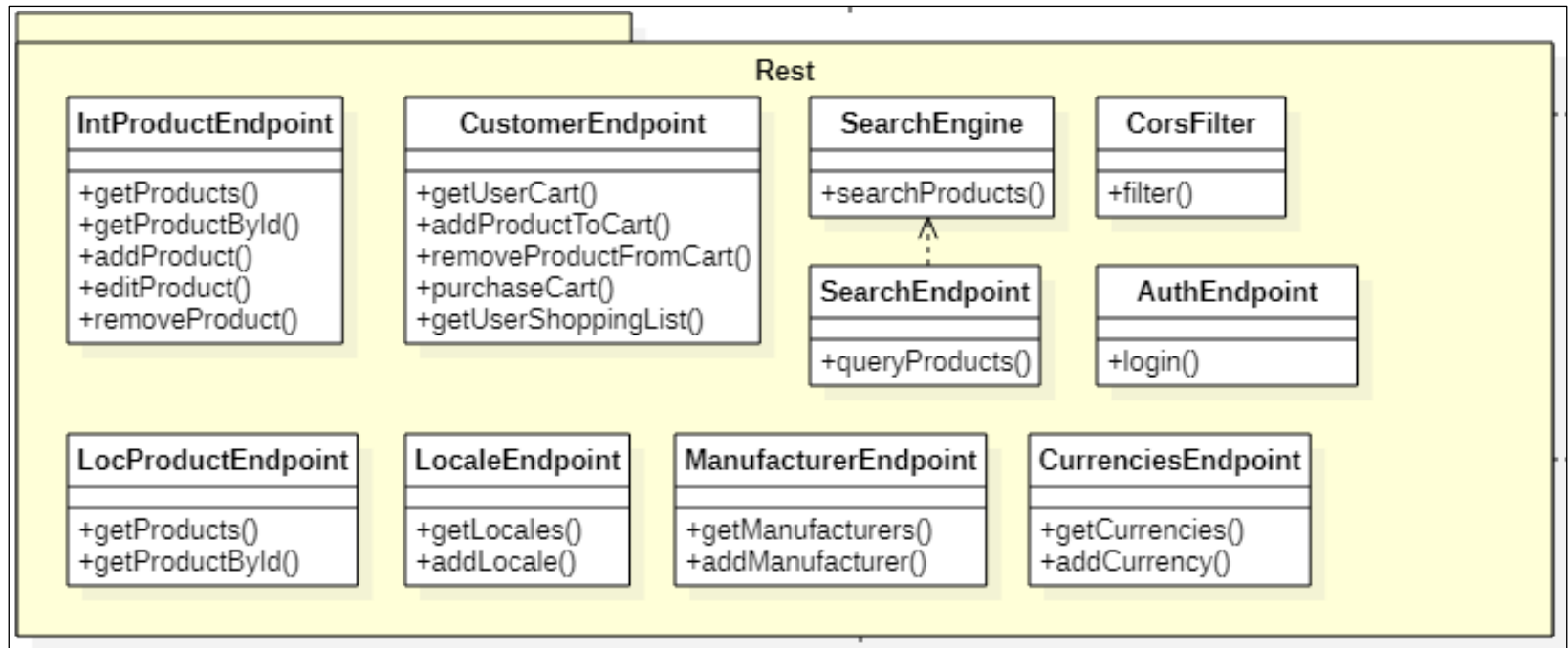


Figure 10: Rest package class diagram

Workflow

Security Package

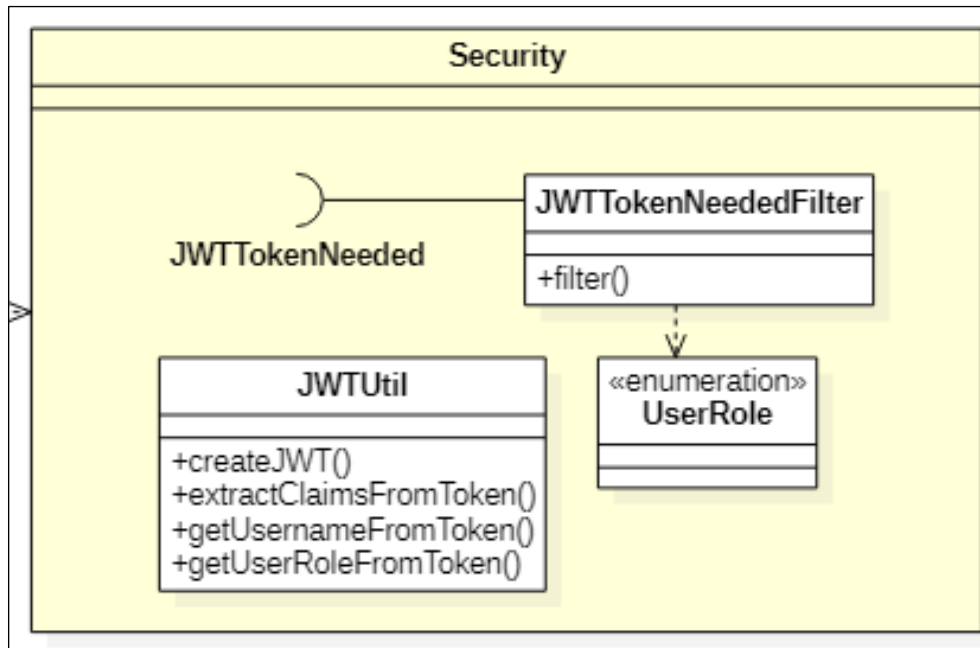


Figure 11: Security package class diagram

```

{
  "subject": "mario.rossi@example.com",
  "issuedAt": 1624223874036,
  "userId": 1,
  "userRole": "ADMIN",
  "lang": "en",
  "exp": 1624224474,
  "issuer": "i18n-store"
}

```

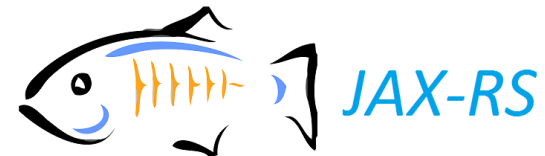
Figure 12: Exchanged Json Web Token [3] payload

Workflow

Endpoints definition

The following groups of REST APIs have been modeled in order to let the clients use the system functionalities:

- Authentication endpoint
- Administration endpoints
- Customer endpoints
- Search endpoint



Each group contains a set of endpoints that allows to manage products (administration) and purchase them (customer). Authentication and search are common functionalities that can be used by both types of users.

Workflow

Endpoints definition

- **GET** `/api/auth/login`: User authentication, sent JWT to user if given username and password are correct
- **GET** `/api/users`: Show all the users
- **GET** `/api/int-products`: Show all the internationalized products (along with all localizations)
- **GET** `/api/int-products/{prodId}`: Show specified internationalized product (along with all localizations)
- **POST** `/api/int-products`: Add an internationalized product
- **PUT** `/api/int-products/id`: Edit an internationalized product
- **DELETE** `/api/int-products/{prodId}`: Remove an internationalized product
- **GET** `/api/locales`: Show all configured locales
- **POST** `/api/locales`: Add a locale
- **GET** `/api/manufacturers`: Show all manufacturer
- **POST** `/api/manufacturers`: Add a manufacturer
- **GET** `/api/currencies`: Show all currencies
- **POST** `/api/currencies`: Add a currency

Figure 3: Authentication endpoint

Figure 3: Administration endpoints

Workflow

Endpoints definition

- **GET** `/api/products`: Show all products, localized in user locale
- **GET** `/api/products/{prodId}`: Show specified product, localized in user locale
- **GET** `/api/customer/{userId}/shopping-cart`: Show user shopping cart
- **POST** `/api/customer/{userId}/shopping-cart/{prodId}`: Add specified product to user shopping cart
- **DELETE** `/api/customer/{userId}/shopping-cart/{prodId}`: Removed specified product from the user shopping cart
- **POST** `/api/customer/{userId}/shopping-cart/checkout`: Purchase all the product in shopping cart and move them to the shopping list
- **GET** `/api/customer/userId/shopping-list`: Show user shopping list
- **GET** `/api/kw-products?query={query}`: Search products according to their name and description with a query (keyword separated by "+" character)
- **GET** `/api/mlt-products?like={id}`: Search products similar to the one specified by the given identifier

Figure 3: Customer endpoints

Figure 3: Search endpoints

Workflow

Sequence diagrams

Through sequence diagram, specific actions of a use case can be modeled in order to describe what messages are exchanged between the entities to accomplish the operation.

The following sequence diagrams have been created:

- Add Product
- Show Product
- Add Product To Cart
- Purchase Products



Workflow

Sequence diagrams

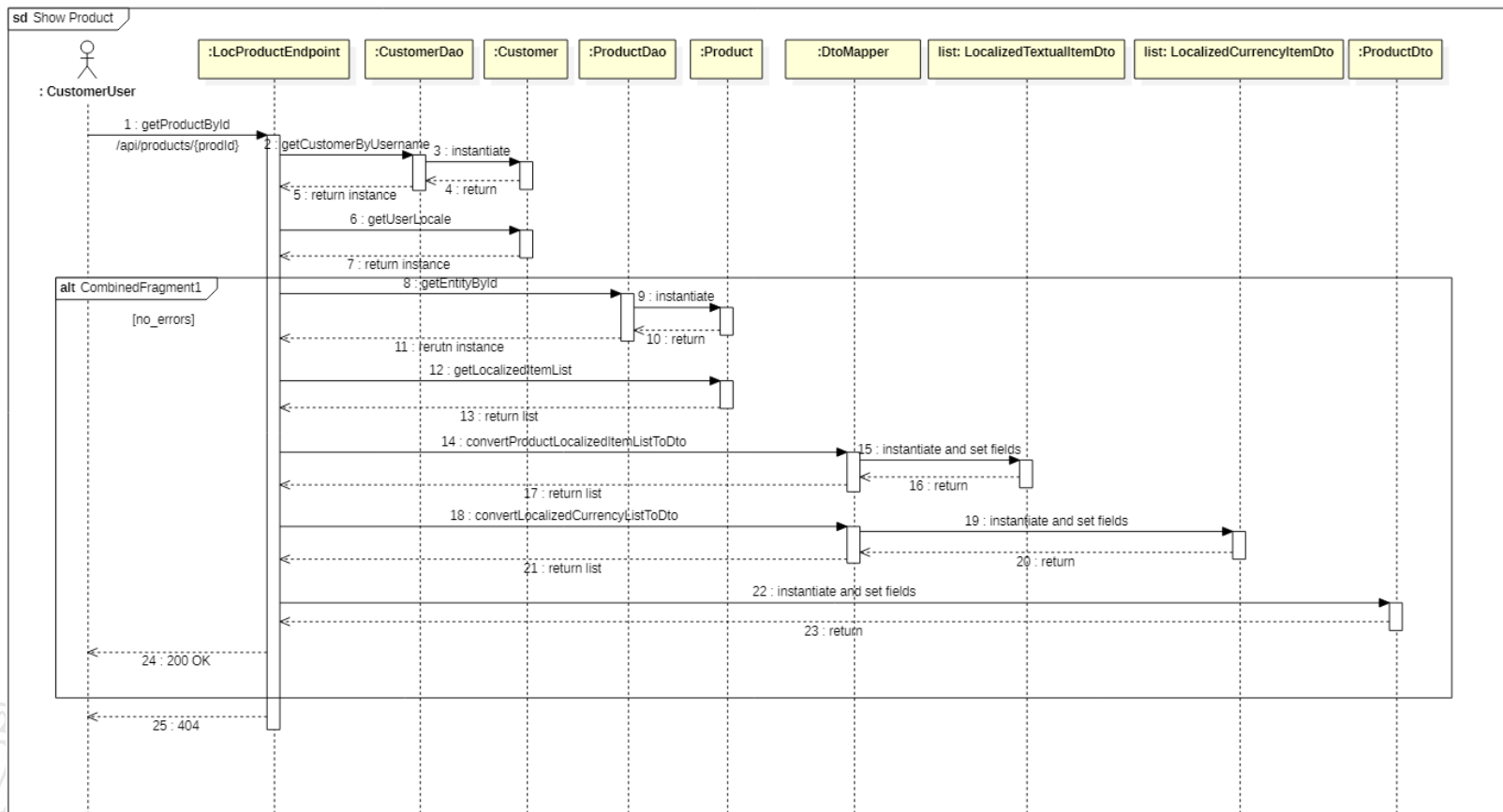


Figure 13: Show product sequence diagram



Workflow

Application development

After the design, the application has been developed by implementing each package using the requested technology stack (JPA, CDI, JAX-RS, Hibernate Search).

The following functionalities are offered by the system:

- **Administration:** add, edit and remove a localized product, manage application locales and currencies, show users and manufacturers, show products with all their localizations;
- **Customer:** show product localized in user locale, purchase products through the use of a shopping cart;
- **Common:** Authentication, product search by keywords and similarity through Hibernate Search.



Workflow

Implemented search functionalities

The following functionalities have been implemented with Hibernate Search:

- **Startup indexing:** a bean is responsible of the indexing made at startup time
- **Standard OR search:** searches on the inverted index using an *OR query*
- **Standard AND search:** searches on inverted index using an *AND query*
- **Fuzzy search:** approximate matching with *Levenshtein distance*
- **Phrase search:** search for exact or approximate *sentences*
- **Similarity search:** search for entities *similar* to the requested one



Workflow

Application testing

Application development has been supported by test units.

Tests have been developed for the following package using the **JUnit** [4] **framework**:

- Model (domain + translation)
- Dao
- Dto
- Security



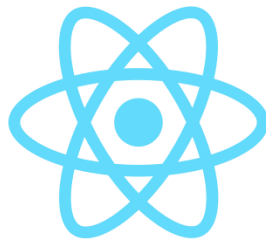
Controller endpoints have been tested through the use of **Postman**, by creating a collection of REST call that can be run to test the correctness of returned values and the error handling



Frontend Application

React Redux

- A demonstrative frontend application has been developed to consume the offered REST APIs and to show the functionalities of the backend application.
- The frontend application has been developed using the **React** [5] framework with the **Redux** [6] library.
- React allows to easily build user interfaces and thanks to Redux the MVC architecture can be implemented.



Frontend Application

Frontend pages

Administration section:

- **Products:** available products and all their configured localizations
- **Users, Locales, Manufacturers, Currencies:** these pages shows the respective configured values

Customer section:

- **Products:** available products, localized in customer locale
- **Shopping Cart:** allows to purchase products
- **Shopping List:** show purchased products

The user interface has been **internationalized** through the use of **i18next** library



Frontend Application

Angular vs React Redux

Angular	React + Redux
<ul style="list-style-type: none">● TypeScript based● Components can communicate by passing data to each others● Components splitted in three parts: HTML (view), TS file (logic), CSS (style)● Various component lifecycle hooks● Data model can be defined thanks to TypeScript● Services as controllers	<ul style="list-style-type: none">● Javascript based● Components can communicate by passing data to each others● Components implement render methods (view) and handle the logic● Various component lifecycle hooks● No standard data model definition in React vanilla. Centralized state with Redux● No standard controller definition in React vanilla. Actions + Dispatchers with Redux



Fronted Application

Demo

I18N Store

Administration

Products

Users

Manufacturers

Locales

Currencies

Log out

Product Informations

Product ID: 1

Edit product

Remove product

Manufacturer: Sony

Locale: en-US

Product Name

Sony Alpha a7II

Description

The Sony Alpha a7II Mirrorless Digital Camera is the world's first full-frame camera with 5-axis image stabilization and provides camera shake compensation for wide-ranging mountable lenses.

Price

1598 \$

Locale: it-IT

Product Name

Sony Alpha a7II

Description

Struttura completa, dimensioni del palmo. Perfezione per tutti. Stabilità per tutti. La qualità delle immagini mozzafiato incontra la libertà di ripresa senza pari nel 7 II, la prima fotocamera full-frame al mondo con stabilizzazione dell'immagine a 5 assi.

Figure 15: Frontend product administration section

Fronted Application

Demo



Figure 16: Frontend product customer section

Conclusions

- The developed prototype application is a shop designed with internationalization features: shop products can be easily localized in multiple languages.
- Thanks to the Hibernate Search library, full-text search feature on these localized items has been implemented. Various types of search can be accomplished, from simple standard searches to similarity searches.
- Application development leaded by design and testing stages.
- A demo frontend application has been developed using React Redux. The frontend application allows to use the main functionalities of the backend.



References

1. Internationalization/Localization:

<https://www.w3.org/International/questions/qa-i18n>

2. Hibernate Search:

https://docs.jboss.org/hibernate/search/5.10/reference/en-US/html_single

3. Json Web Tokens: <https://jwt.io/>

4. Junit: <https://junit.org/junit5/>

5. React: <https://it.reactjs.org/>

6. Redux: <https://react-redux.js.org/>

