I18N Store

Author: Francesco Areoluci

Academic Year: 2020/2021

1. Contesto applicativo

Il progetto si pone come obiettivo la realizzazione di un applicativo rappresentante un negozio che offra la possibilità agli utenti di acquistare prodotti ed effettuarne la ricerca in base al loro nome e alla descrizione, in particolare utilizzando la lingua dell'utente. Tali funzionalità permettono di utilizzare la tecnologia **Hibernate Search** su attributi localizzabili: ogni prodotto avrà dei campi che possono essere disponibili in più lingue, come il nome e la descrizione e tali versioni localizzate apparterranno al medesimo prodotto.

2. Requisiti applicazione

Nei successivi paragrafi sono descritti i requisiti dell'applicazione suddivisi in funzionali (FR), non funzionali (NFR), di dominio (DR). Sono inoltre specificati i vincoli di progetto (C).

2.1 Requisiti funzionali

- **FR1**: Il sistema deve prevedere la possibilità di gestire i prodotti che sono disponibili per la ricerca e l'acquisto
- **FR2**: Ciascun prodotto deve contenere i seguenti campi:
  - **FR2.1**: Nome prodotto
  - **FR2.2**: Descrizione prodotto
  - **FR2.3**: Prezzo
  - **FR2.4**: Produttore
  - **FR2.5**: Categoria di prodotto
- **FR3**: Il sistema deve prevedere la possibilità che i prodotti siano localizzati in più lingue. Tale localizzazione deve avere effetto sui seguenti campi:
  - **FR3.1**: Nome prodotto
  - **FR3.2**: Descrizione prodotto
  - **FR3.3**: Prezzo
  - **FR3.5**: Categoria di prodotto
- **FR4**: Il sistema deve prevedere la possibilità di visionare ed acquistare prodotti.
  - **FR4.1**: La visione dei prodotti deve essere gestita in modo localizzato. Un determinato prodotto deve venir restituito a chi ne fa richiesta con i campi specificati in **FR3** localizzati in modo opportuno.
- **FR5**: Il sistema deve prevedere la possibilità di ricercare prodotti
- **FR6**: Il sistema deve prevedere la possibilità di gestire due tipi di account:
  - **FR6.1**: Account amministratore, con responsabilità di gestione dei prodotti e di ricerca, come specificato rispettivamente in **FR1** e in **FR5**;
  - **FR6.2**: Account acquirente, con responsabilità di acquisto e ricerca, come specificato rispettivamente in **FR4** e **FR5**.

2.2 Requisiti non funzionali

- **NFR1** (Security Requirement): L'accesso all'applicativo deve essere gestito tramite autenticazione utente per entrambe le tipologie di account descritte in **FR6**
- **NFR2** (Implementation Requirement): La funzionalità di acquisto di prodotti, come specificata in **FR4**, deve essere implementata in modo astratto con le seguenti modalità:
    - **NFR2.1**: l'utente deve avere a disposizione un carrello di prodotti con cui può aggiungere o rimuovere i prodotti da acquistare.
    - **NFR2.2**: completata la scelta, l'utente può procedere all'acquisto ed i prodotti acquistati vengono trasferiti dal carrello ad una lista di acquisti.

2.3 Requisiti di Dominio

- **DR1**: La gestione dei prodotti, come specificata in **FR1**, deve prevedere la possibilità di aggiungere, rimuovere e modificare i prodotti disponibili (operazioni CRUD)
- **DR2**: La ricerca dei prodotti, come specificato in **FR5**, deve essere basata su parole chiave e deve essere effettuata sul nome e descrizione dei prodotti
- **DR3**: Ad un account acquirente deve essere associata una lingua per la visione localizzata dei prodotti.

2.4 Vincoli di progetto

- **C1**: L'applicativo deve essere sviluppato in Java EE, sfruttando JPA e CDI
- **C2**: Il layer di persistenza deve essere gestito tramite Hibernate
- **C3**: La persistenza deve essere gestita tramite DBMS MariaDB
- **C4**: Il layer di ricerca deve essere gestito tramite la libreria Hibernate Search
- **C5**: I servizi dell'applicativo devono essere esposti tramite REST API utilizzando la tecnologia JAX-RS
- **C6**: La gestione dei pacchetti deve essere gestita con Maven

3. Casi d'uso

Di seguito sono riportati i casi d'uso per le due tipologie di utente, derivati dai requisiti precedentemente descritti.

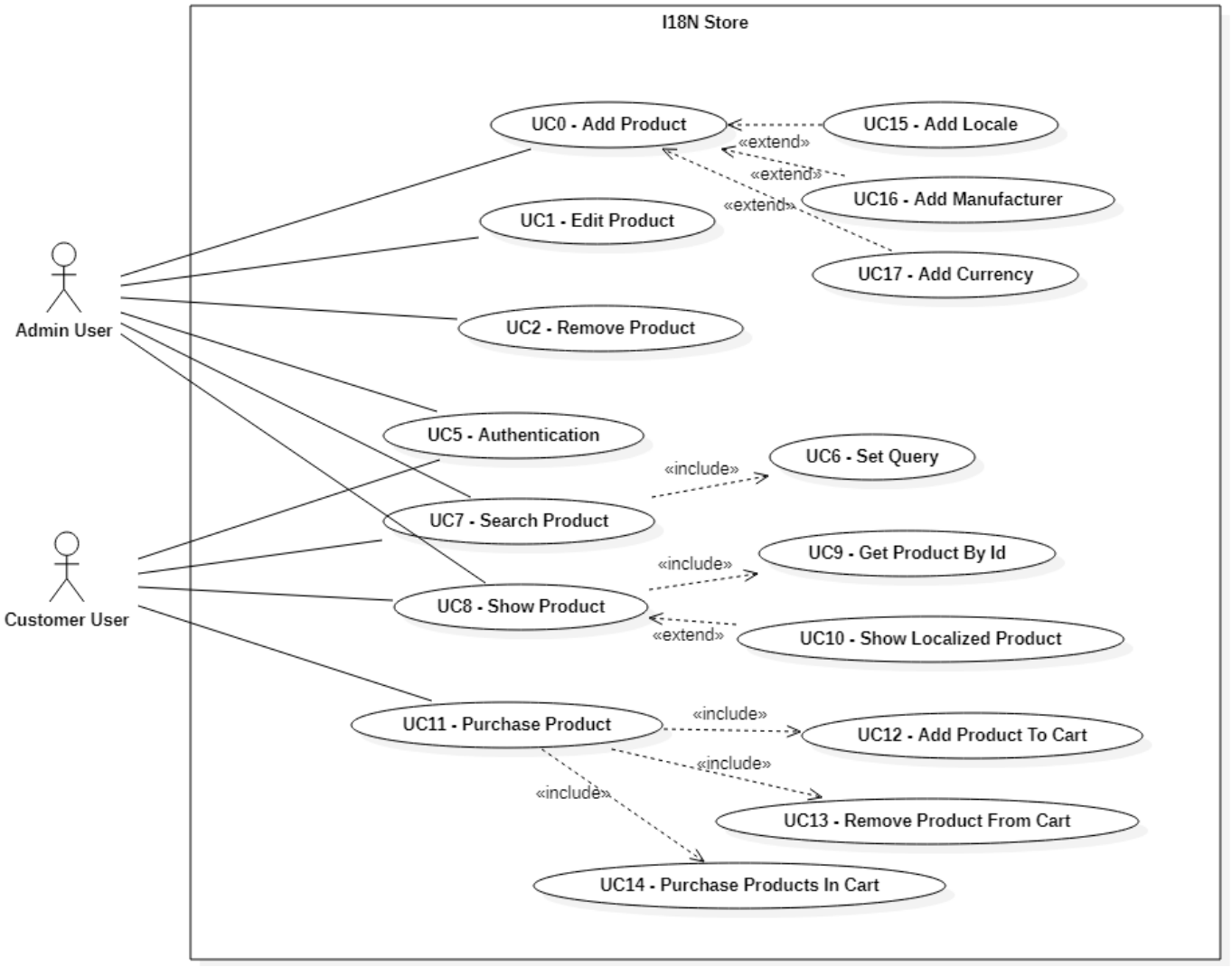

***Figura 1**: Use Case Diagram per tipologia di utente*

Dal diagramma si evince che:

- L'utente amministratore potrà interagire con l'applicativo tramite inserimento, modifica e rimozione di prodotti. Inoltre per inserire i prodotti potrà aggiungere locali, produttori e valute.
- L'utente acquirente potrà interagire con l'applicativo tramite visione (localizzata) ed acquisto dei prodotti.
- I casi d'uso a comune tra i due utenti sono autenticazione, ricerca e visione dei prodotti.

4. Modellazione applicativo

Il seguente class diagram rappresenta l'intera modellazione dell'applicativo. I domini logici sono stati suddivisi in packages. All'interno del diagramma sono presenti i seguenti packages:

- **Modello di dominio**: rappresenta il dominio dell'applicazione;
- **Controllers**: contiene i controllori (endpoint) utilizzabili;
- **Data Access Objects**: contiene le entità responsabili dell'interfacciamento con il DBMS e del relativo instanziamento delle entità del modello di dominio e di traduzione.
- **Data Transfer Objects**: contiene le entità utilizzate per trasferire dati da endpoints a client, in formato JSON.
- **Security**: contiene le entità responsabili della gestione dell'autenticazione e autorizzazione degli utenti.

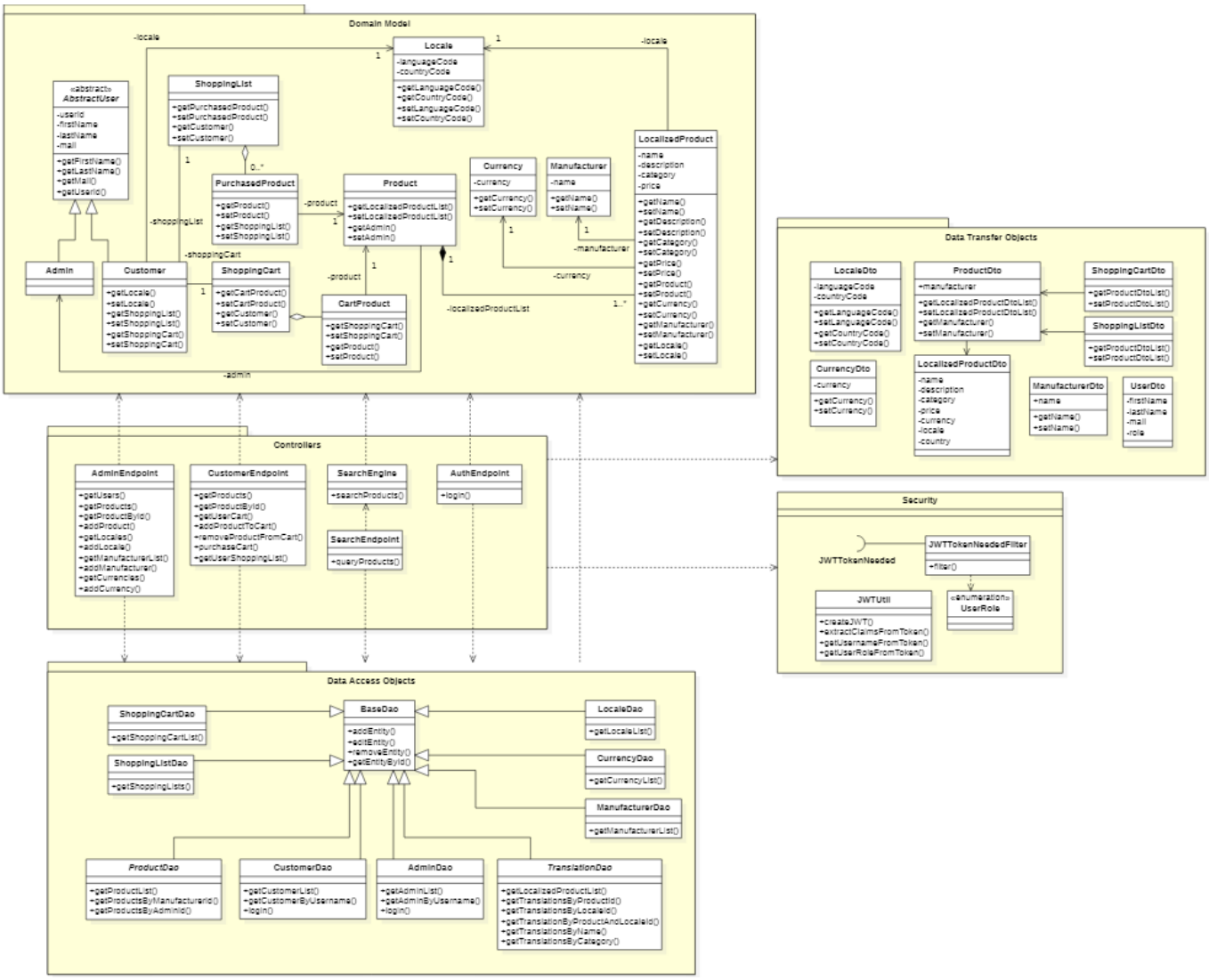

Figura 2: Class Diagram dell'applicativo

Nei prossimi paragrafi saranno esaminati i vari packages.

4.1 Modello di dominio

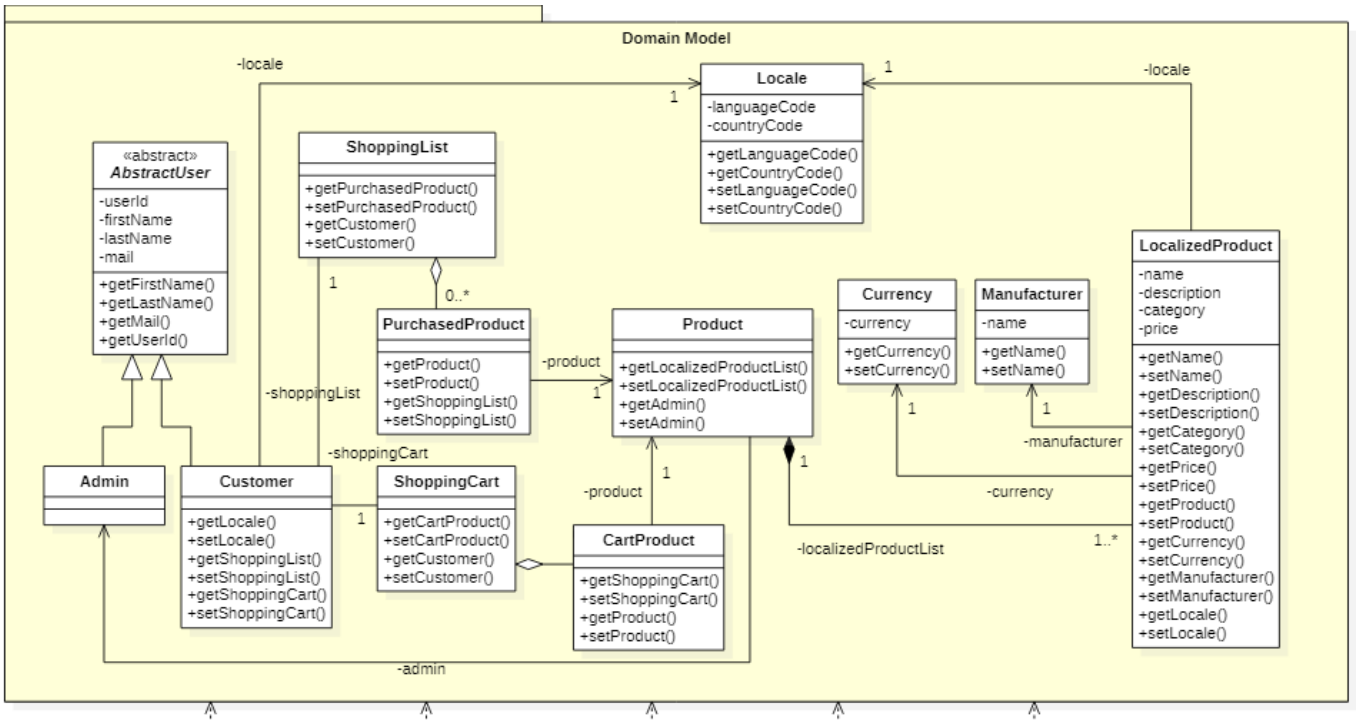

***Figura 3**: Domain Model*

Il modello di dominio rappresenta le entità base del contesto in esame. Contiene principalmente entità POJO. In particolare troviamo:

- le due tipologie di utente (Admin e Customer)
- l'entità prodotto (Product), la cui istanza viene inserita da un Admin e associata a 1 o più localizzazioni di prodotto (LocalizedProduct). Questa entità contiene i campi localizzati in una certa lingua per un certo prodotto.
- il carrello (ShoppingCart) e la lista di prodotti acquistati (ShoppingList). Queste entità sono associate rispettivamente ai prodotti nel carrello (CartProduct) e ai prodotti acquistati (PurchasedProduct)
- i locali associati al Customer e ai LocalizedProduct

Le entità sono instanziate e persistite dai controllori e dalle entità DAO.

4.2 Controllori

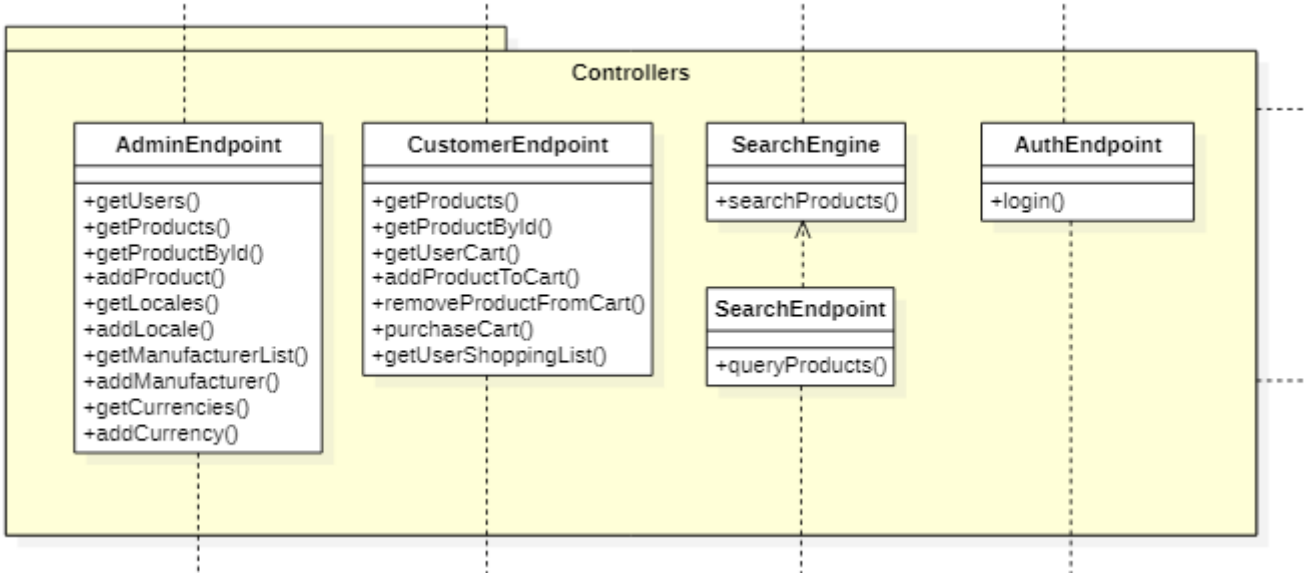

***Figura 4**: Controllori*

I controllori permettono di operare e manipolare le entità del modello di dominio tramite servizi che espongono funzionalità ai client dell'applicativo. In particolare i controllori impiegati in questo layer offrono:

- interfaccia di autenticazione (AuthEndpoint)
- interfaccia per l'amministratore (AdminEndpoint)
- interfaccia per l'acquirente (CustomerEndpoint)
- interfaccia di ricerca (SearchEndpoint) ed entità responsabile per la ricerca dei prodotti (SearchEngine)

I controllori manipolano le entità del modello di dominio utilizzando i Data Access Objects (DAO) messi a disposizione nel layer dedicato.
Questi controllori sono esposti ai client come interfaccia REST, utilizzando la libreria JAX-RS. Lo scambio di dati tra questo layer ed i client viene gestito tramite oggetti JSON, che sono creati a partire da entità base, esposte nel layer Data Transfer Objects (DTO).
L'autenticazione ed autorizzazione è gestita tramite Json Web Token (JWT). Questa tipologia di token permette una gestione stateless dell'autenticazione degli utenti grazie alla verifica di una signature generata da un'entità certificata. L'autenticazione in questo layer è resa possibile utilizzando un'annotazione esposta nel layer Security. **Nota**: è necessario utilizzare HTTPS (SSL/TLS) per usufruire della cifratura della richiesta di login iniziale e del successivo scambio di token.

4.3 Data Access Objects

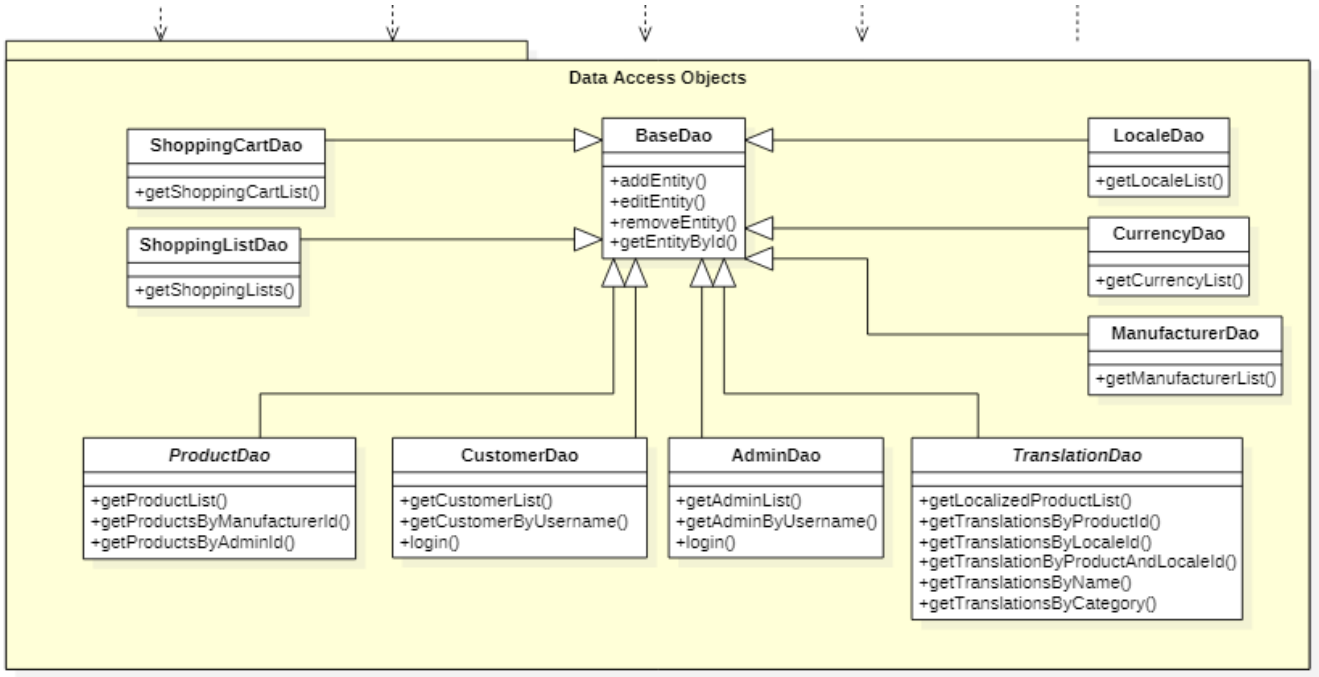

***Figura 5**: DAO*

I Data Access Objects implementano tutte le funzionalità di accesso e persistenza verso il DBMS, astraendo tali funzionalità ed offrendole al layer dei controllori. Le funzionalità a comune tra tutti i DAO sono contenute nell'entità BaseDao e consistono in:

- Ricerca di un'entità per identificativo
- Persistenza di una nuova entità
- Modifica di una entità persistita
- Rimozione di una entità persistita

Le altre entità implementano operazioni specifiche a seconda dell'entità di riferimento.

4.4 Data Transfer Objects

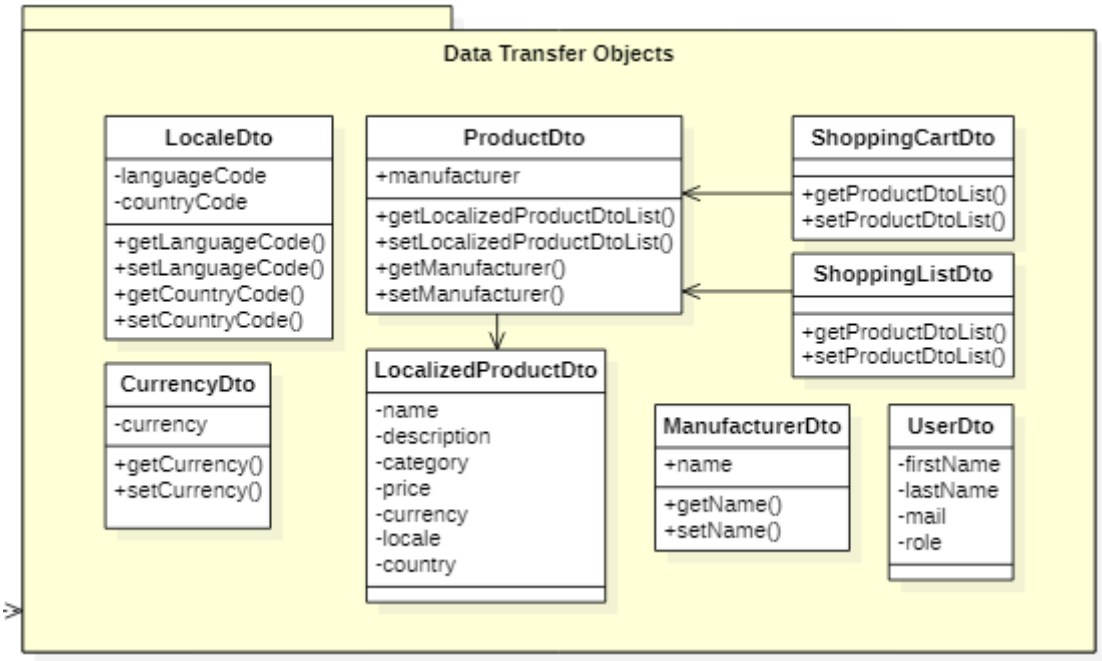

***Figura 6**: DTO*

I Data Transfer Objects sono entità POJO che permettono di modellare oggetti che possono essere spediti o ricevuti come JSON. Sono quindi utilizzati dal layer dei controllori per gestire, in alcuni endpoint, le risposte ed i dati in ingresso. Questo layer è stato utilizzato per astrarre i messaggi di interfacciamento con i client dal modello di dominio.

4.5 Security

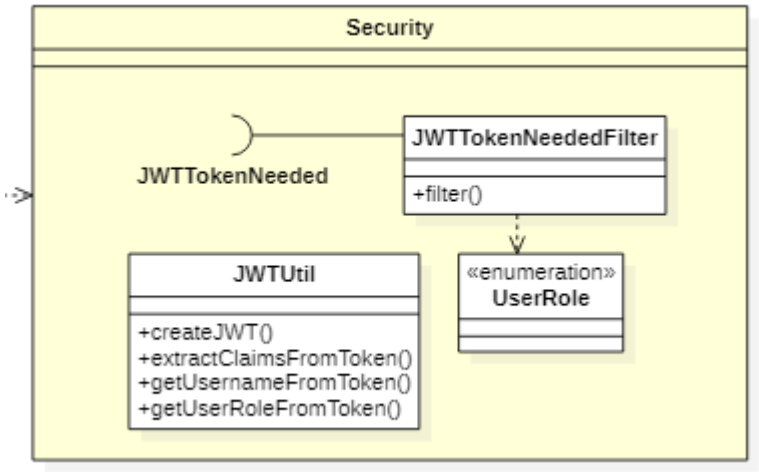

***Figura 7**: Security*

Questo layer implementa le feature di autenticazione ed autorizzazione utilizzando i Json Web Token. I token sono firmati con HMAC + SHA256 e, per semplicità, l'emissione e la validazione sono gestiti dall'applicativo stesso. Questi token hanno il payload strutturato nel seguente modo:

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

L'inclusione del ruolo incluso nel payload permette di gestire anche l'autorizzazione sugli endpoint esposti: gli endpoint relativi all'amministratore sono accessibili soltanto da utenti con ruolo ADMIN, quelli relativi agli aquirenti sono accessibili soltanto da utenti con ruolo CUSTOMER.

5. Endpoint REST

Di seguito sono elencati gli endpoint esposti dall'applicativo.

- Authentication endpoint (**/api/auth/**):
    - **/api/auth/login**: Autenticazione utente, distribuisce JWT se username e password sono corretti
- Admin endpoint (**/api/admin/**):
    - **/api/admin/users**: Visualizza tutti gli utenti
    - **/api/admin/products**: Visualizza tutti i prodotti
    - **/api/admin/products/{id}**: Visualizza prodotto specifico, con tutte le traduzioni
    - **/api/admin/products/add**: Aggiunge prodotto
    - **/api/admin/products/remove/{id}**: Rimuove prodotto
    - **/api/admin/locales**: Visualizza tutti i locale configurati
    - **/api/admin/locales/add**: Aggiunge locale
    - **/api/admin/manufacturers**: Visualizza tutti i produttori
    - **/api/admin/manufacturers/add:** Aggiungi un produttore
    - **/api/admin/currencies**: Visualizza tutte le valute
    - **/api/admin/currencies/add**: Aggiunge una valuta
- Customer endpoint (**/api/customer/**):
    - **/api/customer/products**: Visualizza tutti i prodotti, localizzati nella lingua utente
    - **/api/customer/products/{id}**: Visualizza prodotto specificato, localizzato nella lingua utente
    - **/api/customer/shopping-cart**: Visualizza carrello utente
    - **/api/customer/shopping-cart/add/{id}**: Aggiunge prodotto specificato a carrello utente
    - **/api/customer/shopping-cart/remove/{id}**: Rimuove prodotto specificato da carrello utente
    - **/api/customer/shopping-cart/checkout**: Aquista i prodotti nel carrello (sposta su ShoppingList)
    - **/api/customer/shopping-list**: Visualizza prodotti acquistati
- Search endpoint (**/api/search/**)
    - **/api/search/products/{query}**: Ricerca prodotti in base a nome e descrizione tramite query (keyword separate da carattere '+')

6. Diagrammi di sequenza

Di seguito sono riportati dei diagrammi di sequenza che espongono le procedure di esecuzione di alcuni casi d'uso specificati nella sezione **3** e che possono essere implementati con la modellazione descritta nella sezione **4**.

6.1 Inserimento prodotto

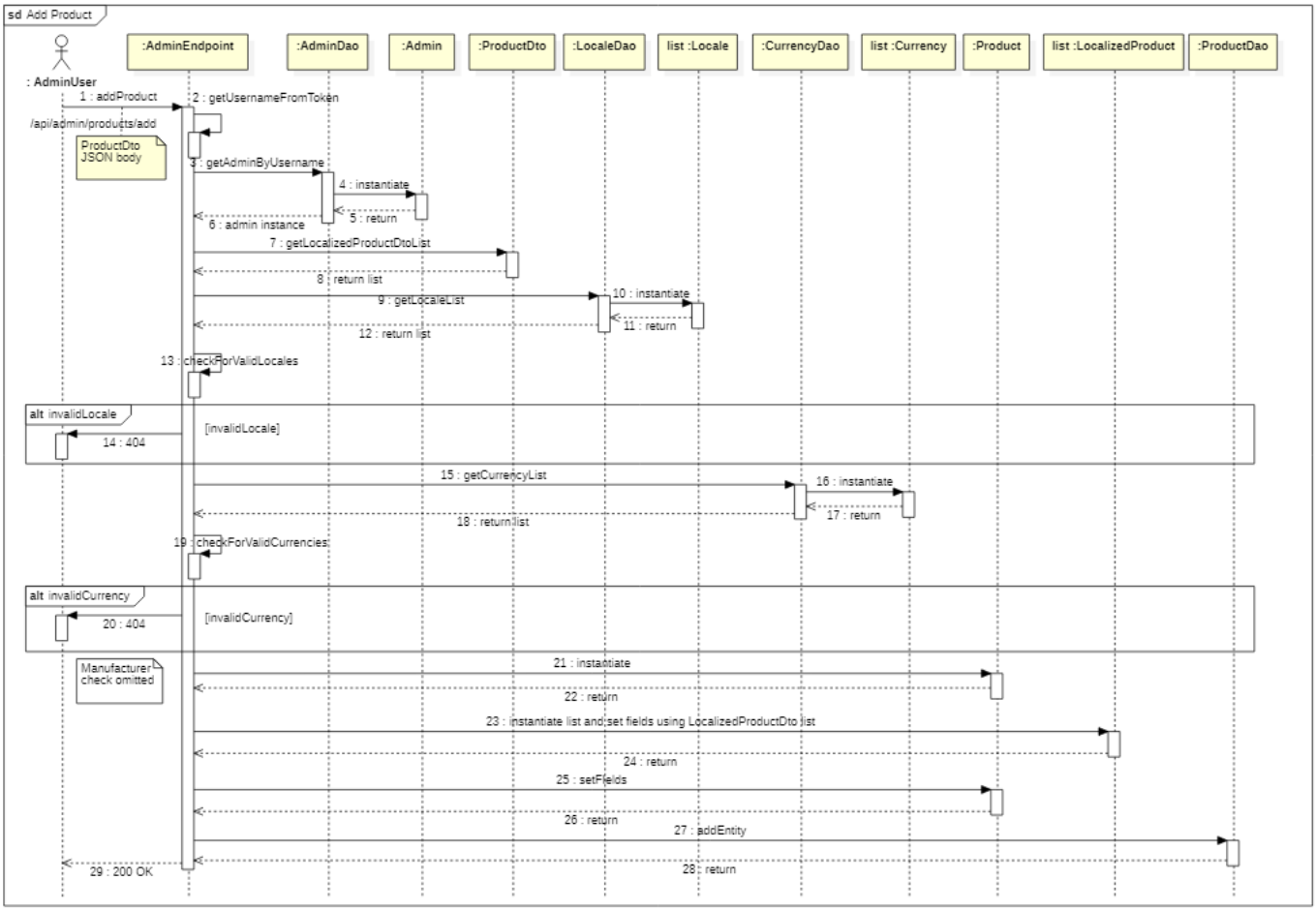

***Figura 8**: Sequence Diagram di inserimento prodotto*

6.2 Visione prodotto

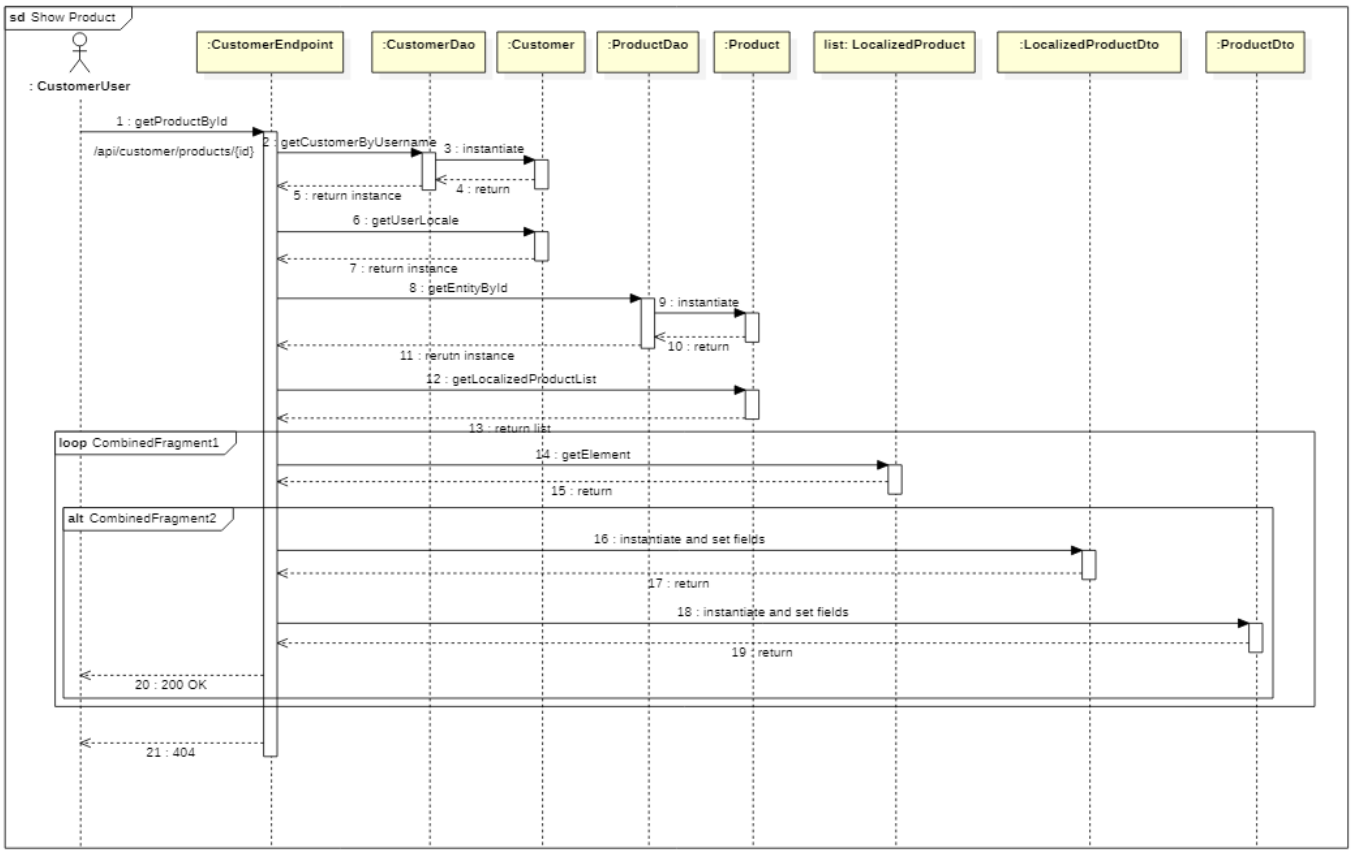

***Figura 9**: Sequence Diagram di visione prodotto*

6.3 Acquisto prodotto

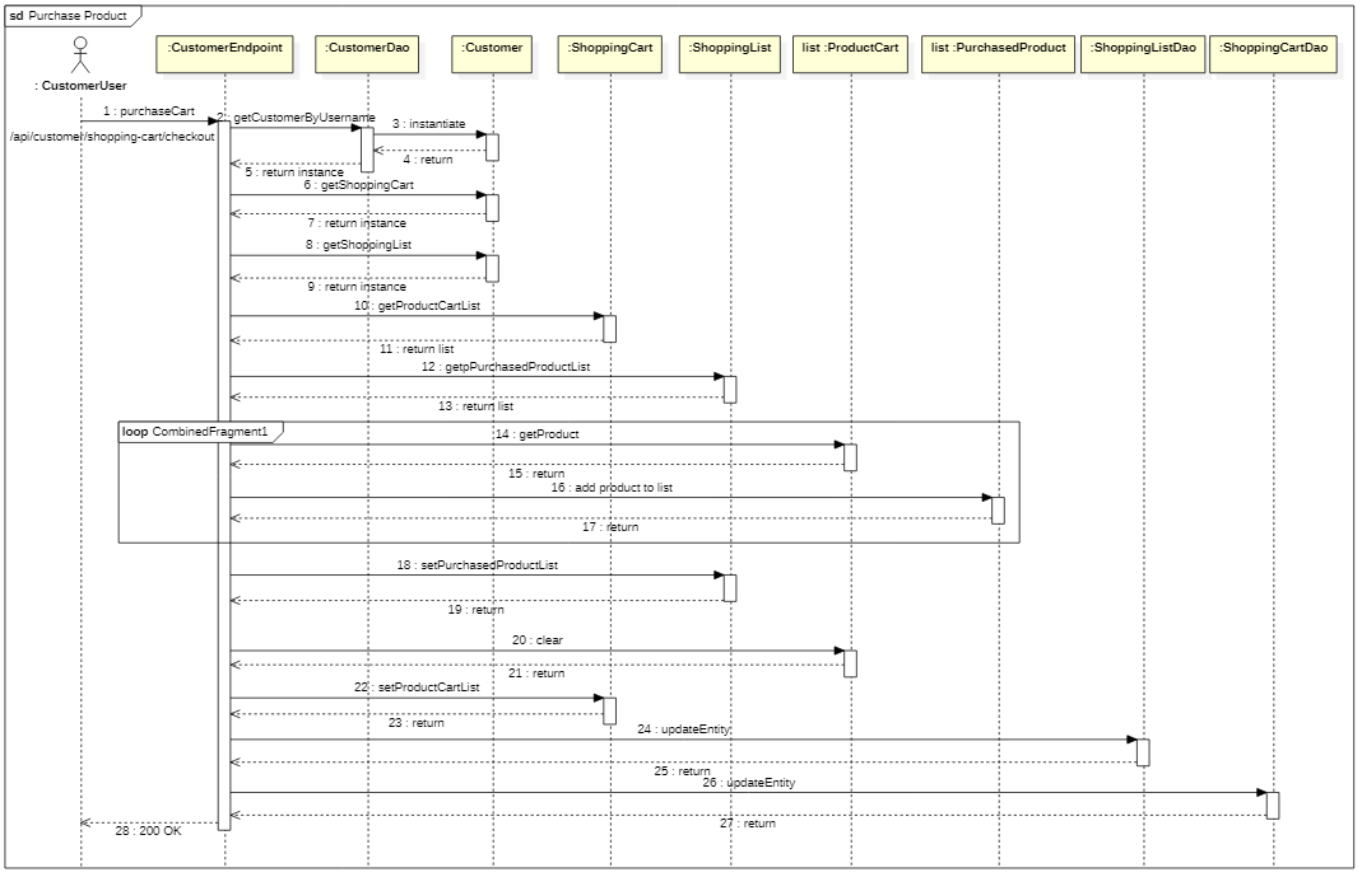

***Figura 10**: Sequence Diagram di acquisto prodotto*

6. Modello E-R

La seguente immagine rappresenta una possibile implementazione del modello su database relazionale a supporto dell'applicativo:

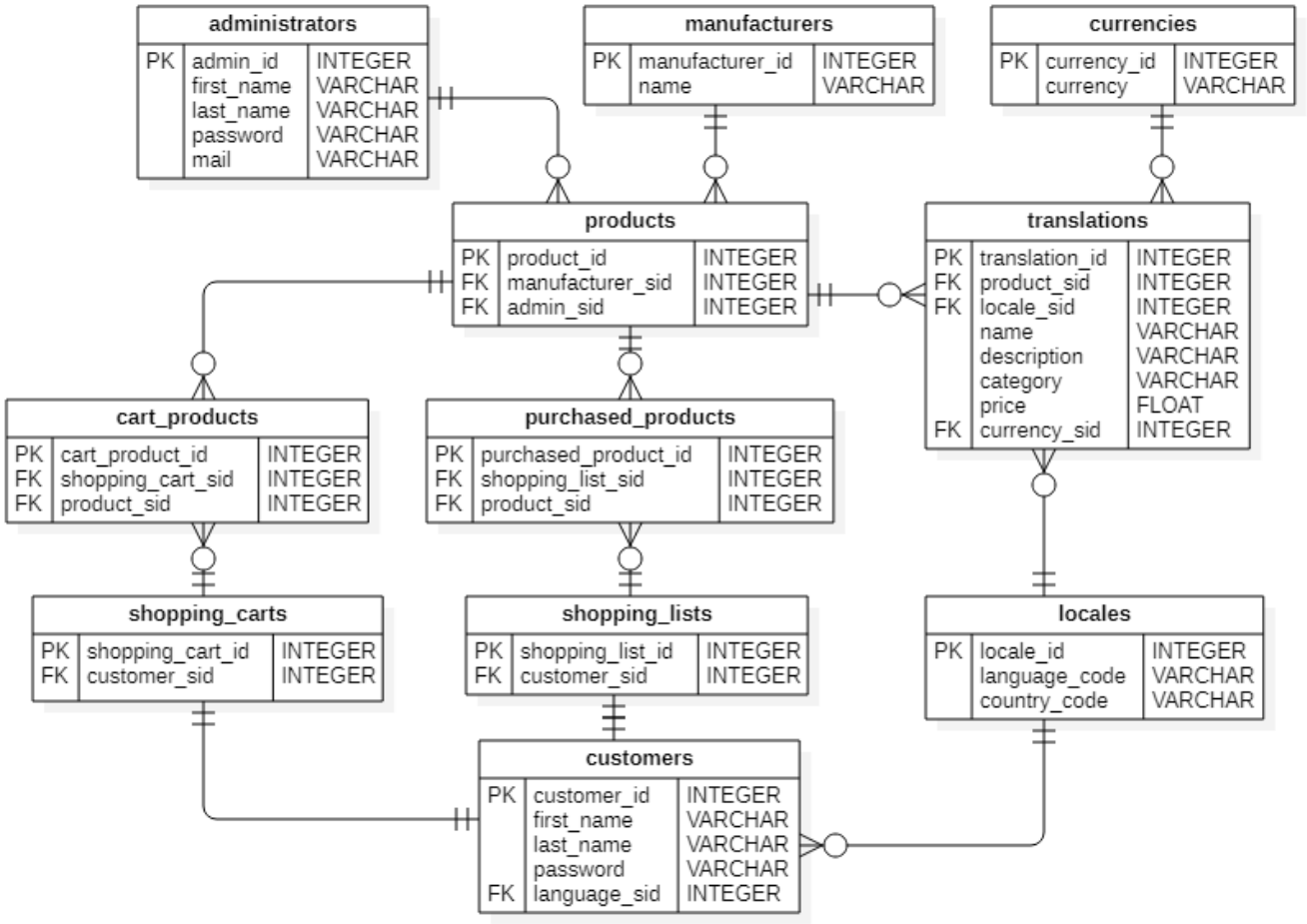

***Figura 11**: Modello ER*

Il modello è formato dalle seguenti entità:

- **administrators**: utenti amministratori che gestiscono i prodotti come specificato nei casi d'uso **UC2**, **UC3**, **UC4**;
- **customers**: utenti acquirenti, che gestiscono un carrello di prodotti (**shopping_cart**) ed hanno una lista di prodotti acquistati (**shopping_list**). Sono associati ad un determinato **locale**.
- **cart_products**: relativi ad un certo carrello e ad un certo prodotto.
- **purchased_products**: relativi ad una shopping list e ad un certo prodotto.
- **products**: prodotti disponibili, associati ad un **manufacturer** ed inseriti da un **administrator**. La localizzazione di un prodotto viene gestita tramite associazione all'entità **translations**.
- **translation**: le traduzioni sono separate dall'entità **products** ed ogni traduzione è associata ad un certo **locale**. In tal modo, la possibile aggiunta di un nuovo locale a supporto dell'applicazione risulterebbe nell'aggiunta di nuove righe in questa tabella, piuttosto che nell'aggiunta di campi aggiuntivi nella tabella **products**. Inoltre una traduzione è associata ad una particolare valuta (**currencies**) per il costo del prodotto.

7. Testing dei moduli

A supporto dell'applicativo è stata creata una test suite che permette di testare sia i singoli packages (JUnit), sia di effettuare test di integrazione valutando il corretto funzionamento di tutti i moduli (Postman).
Tramite JUnit sono stati creati dei test case per i seguenti packages:

- Model
- Dao
- Dto
- Security

Il testing dei controller e di integrazione è stato invece realizzato tramite Postman, con il quale è stata creata una collezione che permette di effettuare tutte le chiamate REST verso gli endpoint esposti, per poter verificare sia la correttezza dei valori ritornati, sia la corretta gestione degli errori.

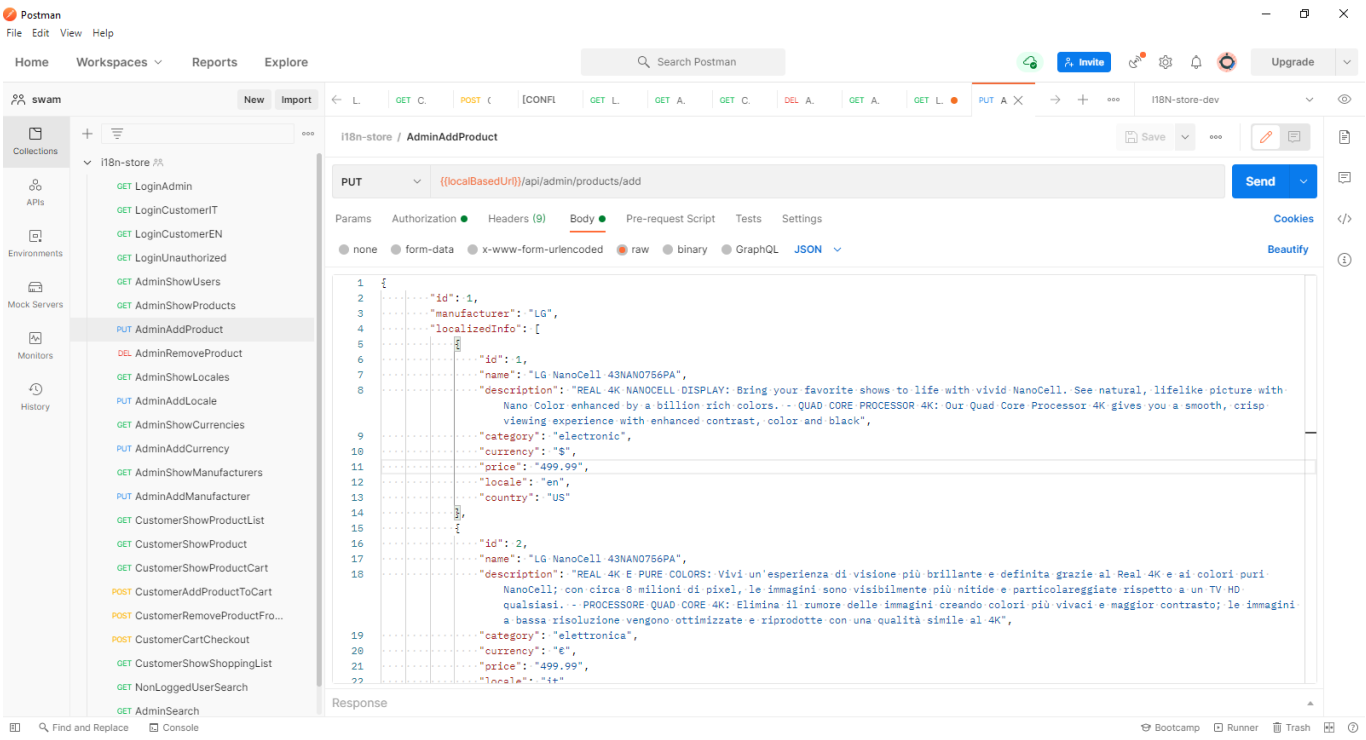

***Figura 12**: Postman*