Projecte d'Intercanvi de Llibres

Wellington Stephen Baéz Ramírez May Castells Raga
Cristina Malena Díaz Krishna Ocaña Zevallos
Ainhoa Pérez García Anna Marín Nuño

Maig 2025

Índex

Decisions de Disseny per a la Implementació de Marcatge Semàntic Web 3.0

0. Enllaç al Repositori

`https://github.com/Krisoc123/ProjecteWeb.git`

1. Introducció

La transformació de la nostra web (2.0) Django d'intercanvi de llibres cap a una aplicació Web 3.0 preten millorar la visibilitat en motors de cerca i proporcionar dades estructurades que facilitin la integració amb altres plataformes. El nostre sistema permet als usuaris intercanviar, vendre o donar llibres mitjançant un sistema de punts, i s'ha intentat afegir marcatge semàntic als atributs que representen les entitats ja existents en el model relacional.

1.1 Selecció del Vocabulari Schema.org

Seguint els requisits de l'enunciat, s'ha utilitzat RDFa (Resource Description Framework in Attributes) amb el vocabulari schema.org per implementar el marcatge semàntic. Schema.org ofereix una estructura ben definida per a productes, ofertes comercials i ressenyes d'usuaris, que s'ajusta perfectament al domini de llibres. A més, aquest vocabulari és reconegut directament per Google, Bing i altres motors de cerca principals.

La implementació del marcatge es basa en una entitat principal `Product` amb el tipus addicional `Book`, que serveix com a contenidor per tota la informació semàntica:

```
<div class="book-entry-container" vocab="http://schema.org/" typeof="Product">
    <link property="additionalType" href="http://schema.org/Book" />
```

Això permet que els motors de cerca entenguin immediatament que estem descrivint un producte que específicament és un llibre, heretant totes les propietats tant de Product com de Book.

1.2 Filtres de Template Personalitzats

S'ha integrat el marcatge semàntic amb els models Django existents sense modificar massa codi ni estructura. S'ha creat un sistema de filtres de template personalitzats que converteixen entre les dades del model i els formats requerits per schema.org, localitzats a `templatetags/book_filters.py`.

El filtre `get_condition_schema`, per exemple, transforma l'estat dels llibres del model `Have` en URLs estàndard de schema.org:

```
@register.filter
def get_condition_schema(status):
    condition_map = {
        'new': 'http://schema.org/NewCondition',
        'used': 'http://schema.org/UsedCondition',
        'damaged': 'http://schema.org/DamagedCondition'
    }
    return condition_map.get(status, 'http://schema.org/UsedCondition')
```

Altres filtres implementats inclouen:

- `calculate_availability`: Determina l'estat de disponibilitat basant-se en l'inventari actual
- `format_price_currency`: Adapta el sistema de punts a la representació monetària de schema.org

D'aquesta manera, no ha calgut modificar els models ni gairee les variables de context de les vistes.

1.3 Entitat Product: Propietats i Jerarquia

S'ha decidit d'utilitzar `Product` com a entitat base amb `Book` com a especialització mitjançant `additionalType` es basa en l'herència de propietats de schema.org. Aquesta estructura permet combinar les propietats comercials de Product (ofertes, preus, ressenyes) amb les propietats específiques dels llibres (ISBN, autor, data de publicació).

2. Marcatge Semàntic de Product

Les propietats heretades de `Product` que s'utilitzen en la implementació:

- `name`: Títol del llibre (visible a la interfície)
- `offers`: Informació de venda i disponibilitat
- `review`: Ressenyes individuals dels usuaris
- `aggregateRating`: Valoració mitjana calculada
- `additionalProperty`: Propietats personalitzades per a funcionalitats específiques
- `image`: Portada del llibre obtinguda des d'APIs externes

Implementació del marcatge semàntic per a Product:

```
<div class="book-entry-container" vocab="http://schema.org/" typeof="Product">
    <link property="additionalType" href="http://schema.org/Book" />
    <h1 property="name">{{ mybook.title }}</h1>
    <img property="image" src="https://covers.openlibrary.org/b/isbn/{{ mybook.ISBN }
    <meta property="productID" content="isbn:{{ mybook.ISBN }}">
</div>
```

2.1 Marcatge de *Book*

Les propietats específiques de `Book` que s'afegeixen al marcatge inclouen informació bibliogràfica estàndard:

- `author`: Autor del llibre (modelat com a entitat Person)
- `isbn`: Número ISBN estàndard per identificació única
- `genre`: Temàtica o categoria del llibre
- `datePublished`: Data de publicació original
- `description`: Descripció del contingut (quan està disponible)

Com que la majoria de propietats ja es trobaven en el context de la vista ha estat relativament senzill. Això sí, s'ha hagut d'afegir al context la data de publicació.

```
<p property="author" typeof="Person">
    <span property="name">{{ mybook.author }}</span>
</p>
<h2><span property="genre">{{ mybook.topic }}</span></h2>
<meta property="isbn" content="{{ mybook.ISBN }}">
<meta property="datePublished" content="{{ mybook.publish_date|date:'Y-m-d' }}">
{% if mybook.description %}
```

```
<p class="description" property="description">{{ mybook.description }}</p>
{% endif %}
```

La propietat `productID` utilitza el format estàndard "isbn:" per identificar el llibre de manera única.

3. Sistema de Reviews: Agregats i Individuals

La implementació del sistema de ressenyes combina dos tipus d'entitats schema.org per oferir informació qualitativa completa sobre els llibres:

3.1 Review Individual

Cada ressenya individual utilitza l'entitat `Review` amb les següents propietats específiques:

- `author`: El revisor com a entitat Person amb nom i perfil d'usuari
- `reviewBody`: El contingut textual de la ressenya, validat per longitud i contingut apropiat
- `reviewRating`: Valoració numèrica de 1 a 5 estrelles
- `datePublished`: Data de creació de la ressenya
- `itemReviewed`: Referència al llibre que s'està valorant

La valoració numèrica de les ressenyes s'ha implementat específicament per aquesta entrega (en l'entrega 2 no hi era). Però així podíem fer un *AggregateRating* i tenir un *markup* més complet.

Implementació del marcatge semàntic per a Review:

```
{% for review in reviews %}

```html
<div class="review-card" property="review" typeof="Review">
    <div class="review-header">
        <div class="review-author" property="author" typeof="Person">
            <span property="name">{{ review.user.name }}</span>
        </div>
        <div class="review-rating" property="reviewRating" typeof="Rating">
            <meta property="ratingValue" content="{{ review.rating }}">
            <meta property="bestRating" content="5">
            <meta property="worstRating" content="1">
        </div>
    </div>
    <p class="review-content" property="reviewBody">{{ review.text }}</p>
    <meta property="datePublished" content="{{ review.date|date:'c' }}">
</div>
```

3.2 AggregateRating per a Valoració Global

L'entitat `AggregateRating` proporciona quantitativa calculada dinàmicament a partir de totes les reviews individuals. Aquesta entitat és especialment important per als motors de cerca, ja que permet capturar de manera fàcil la valoració mitjana d'un llibre i el nombre total de ressenyes.
```

Propietats clau de l'AggregateRating:

- `ratingValue`: Mitjana aritmètica de totes les valoracions individuals
- `reviewCount`: Nombre total de ressenyes, que indica la fiabilitat de la valoració mitjana
- `bestRating`: Valor màxim possible (5), necessari per contextualitzar l'escala
- `worstRating`: Valor mínim possible (1), completant l'escala de valoració

El càlcul es realitza a la vista Django:

```
# Càlcul de la valoració mitjana
if reviews.exists():
    avg_rating = reviews.aggregate(Avg('rating'))['rating__avg']
    review_count = reviews.count()
```

Després, s'inclou en el marcatge semàntic de la pàgina del llibre:

```
{% if avg_rating %}
<div property="aggregateRating" typeof="AggregateRating">
    <meta property="ratingValue" content="{{ avg_rating }}">
    <meta property="reviewCount" content="{{ reviews|length }}">
    <meta property="bestRating" content="5">
    <meta property="worstRating" content="1">
</div>
{% endif %}
```

4. Entitat Offer: Comerç i Disponibilitat

L'entitat `Offer` gestiona la informació de venda i compra adaptada al sistema de punts de la plataforma. Cada llibre disponible té una oferta principal que inclou:

- `availability`: Estat de disponibilitat (obtingut segons la taula del model relacional `Have`) (InStock quan hi ha còpies disponibles, OutOfStock quan no)
- `price`: Preu base en punts
- `priceCurrency`: Moneda personalitzada "POINTS" (per al sistema de punts)
- `itemCondition`: Estat físic del llibre (New, Used, Damaged)
- `seller`: Persona que posseeix el llibre i l'ofereix per a l'intercanvi (primer propietari disponible)
- `priceValidUntil`: Validesa de l'oferta, establerta per defecte a un any (per omplir amb les especificacions de schema.org)
- `inventoryLevel`: Nombre exact de còpies disponibles

4.1 Implementació del Marcatge Offer

```
<div property="offers" typeof="Offer">
    <meta property="availability" content="{{ book_availability }}">
    <meta property="price" content="{{ mybook.base_price }}">
    <meta property="priceCurrency" content="POINTS">
    <meta property="itemCondition" content="{{ book_condition|get_condition_schema }}
    <meta property="priceValidUntil" content="{% now 'Y-m-d'|add_years:1 %}">
    <meta property="inventoryLevel" content="{{ available_copies }}">
```

```
    {% if book_sellers %}
    <div property="seller" typeof="Person">
        <meta property="name" content="{{ book_sellers.0.user.name }}">
        <meta property="email" content="{{ book_sellers.0.user.email }}">
        <meta property="address" content="{{ book_sellers.0.user.location }}">
    </div>
    {% endif %}
</div>
```

4.2 Marcatge de Intercanvi

Per representar les funcionalitats d'intercanvi que no tenen equivalent directe en schema.org, s'utilitzen entitats `PropertyValue` personalitzades:

```
{% if available_copies > 0 %}
<div property="additionalProperty" typeof="PropertyValue">
    <meta property="name" content="exchangeAvailable">
    <meta property="value" content="true">
    <meta property="description" content="Book exchange options available">
</div>
{% endif %}
```

4.3 Indicadors de Popularitat

Mapeig amb Models Django:

L'sistema d'intercanvi es basa en tres models principals:

- **Exchange**: Registres d'intercanvis realitzats
- **SaleDonation**: Transaccions de compra-venda amb punts

Aquestes entitats permeten calcular la popularitat dels llibres per al marcatge semàntic.

S'utilitza la propietat `transactionCount` per indicar el nombre total d'intercanvis i vendes associats a un llibre:

```
{% if transaction_count > 0 %}
<div property="additionalProperty" typeof="PropertyValue">
    <meta property="name" content="transactionCount">
    <meta property="value" content="{{ transaction_count }}">
</div>
{% endif %}
```

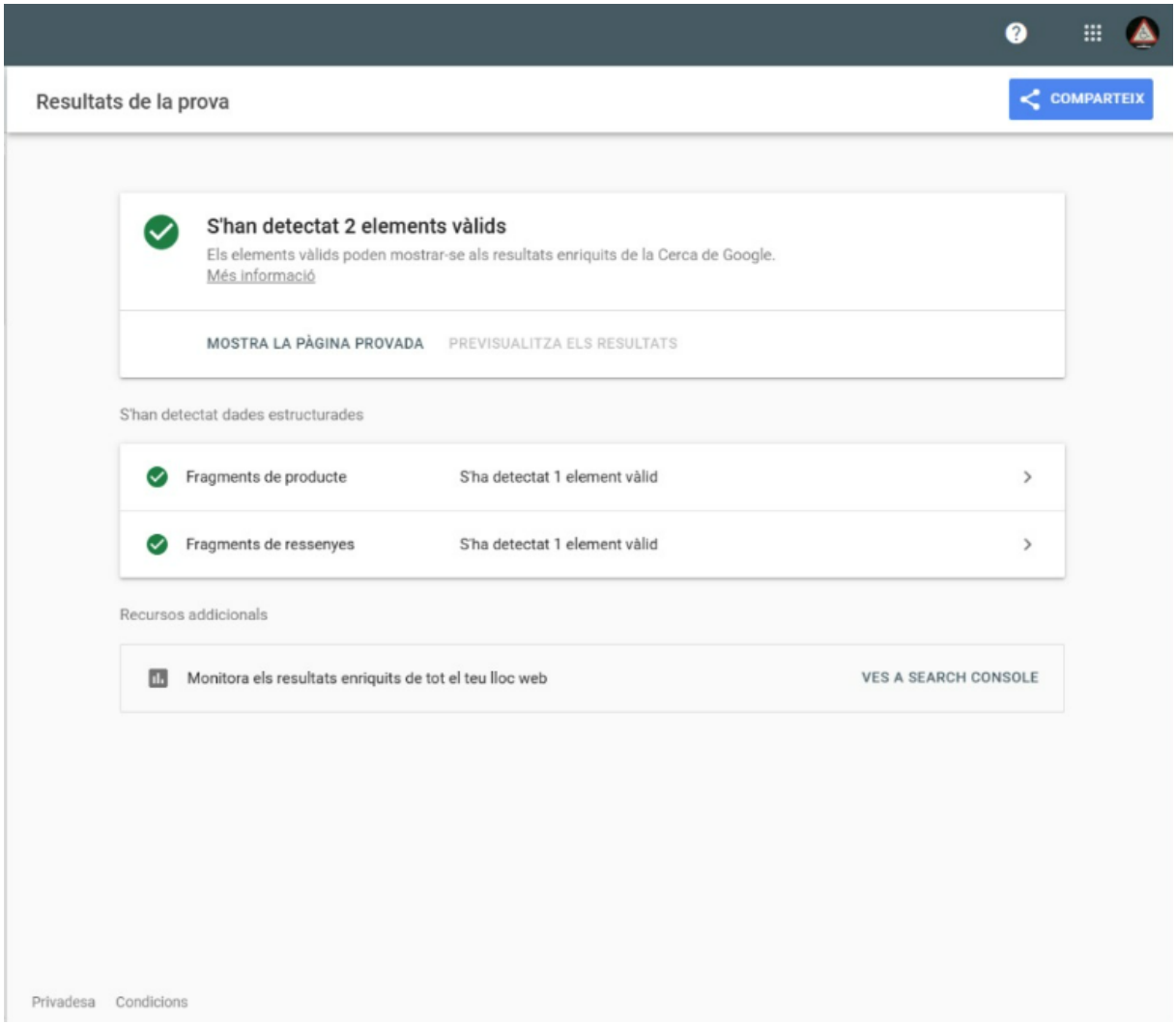
5. Interconexió d'Entitats Schema.org

La implementació final crea com una especie deuna xarxa interconnectada d'entitats que reflecteix amb precisió el model de negoci de la plataforma d'intercanvi de llibres:

1. **Product/Book** actua com a entitat central que conté tota la informació bibliogràfica
2. **Offer** proporciona informació comercial amb propietats adaptades al sistema de punts

3. **AggregateRating** i **Review** ofereixen informació qualitativa i quantitativa sobre els llibres
4. **Person** (autors i reviewers) entitats que representen els usuaris
5. **PropertyValue** elements personalitzats per a funcionalitats d'intercanvi

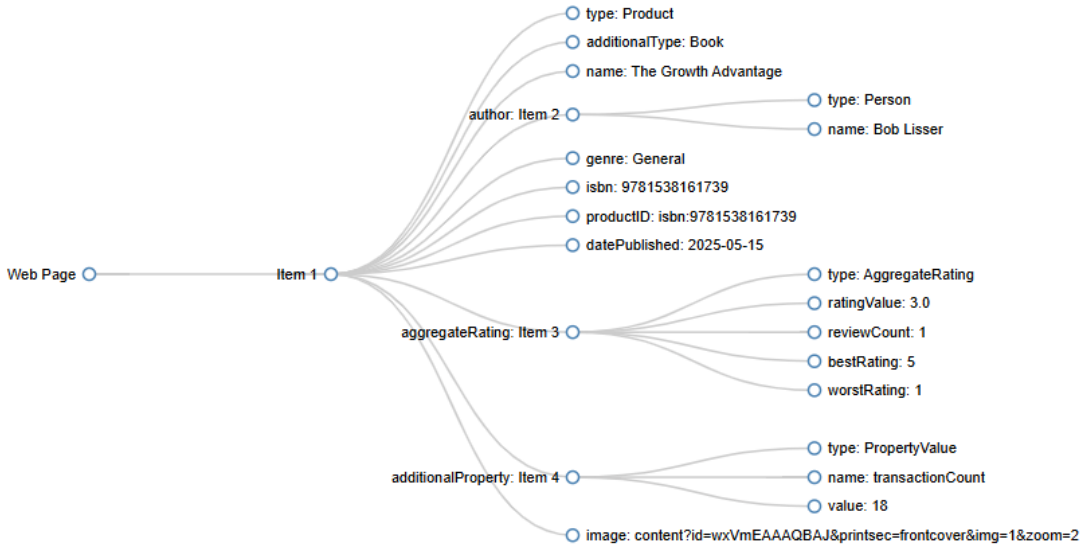
6. Els resultats del test de Google



- Bàsicament, el test de Google ha detectat Products i Reviews.

7. RDFa Play

L'esquema obtingut a partir de l'RDFa Play per un llibre en concret és el següent:



8. Resumint

En aquesta entrega hem buscat totes les entitats que tinguessin alguna relació amb entitats de schema.org i hem intentat relacionar els atributs originals del model Django amb les propietats de schema.org.

També vam començar provant d'afegir marcatge a altres *templates* com ara el de `trending.html`, però vam veure que allà no s'hi mostraven la majoria de propietats que s'esperaven i per no es detectava al test de Google.

A part, en aquesta entrega hem implementat i arreglat algunes coses que no es van fer a l'entrega 2 (ja que pensàvem que potser ens tocaria implementar-les en la següent), com per exemple el sistema de valoracions. També, s'ha corregit alguns errors en l'intercanvi ja que en la base de dades no quedava constància de qui intercanviava cada llibre.

També hem resolt el problema dels punts, per defecte tot usuari tenia 0 punts i no es podia fer cap intercanvi. Ara, cada usuari té 100 punts per defecte i pot comprar llibres.

Finalment, s'ha acabat d'implementar correctament la funcionalitat de venda i donació de llibres.