

Project Name:

Currency Conversion and Transaction History Web Application

Project Goals:

Develop a web application that:

1. Performs currency conversion using up-to-date exchange rates.
 2. Displays a history of completed transactions with filtering options.
 3. Demonstrates practical use of modern frontend and backend technologies.
-

Functionality Description:

1. Currency Conversion:



The screenshot shows a web application titled "Django - Angular convertor". It features two tabs: "Make conversion" (active) and "History access". Under the "Make conversion" tab, there is a form with the following elements: an "Amount" input field containing "\$1", a "From" dropdown menu showing the US flag and "USD US Dollar", a circular currency selector in the center, and a "To" dropdown menu showing the EU flag and "EUR Euro". A blue "View" button is located at the bottom right of the form area.

Ilustración 1 Converter page view

- The user interface is implemented as a single-page application using Angular.
- Currency selection is made through a custom component that includes:
 - Country flags, currency names, and symbols.
 - Dynamic filtering of the currency list based on user input.
 - Component replacement based on user focus, managed with Angular directives (*ngIf).
- Exchange rate data is fetched from a Django-based backend server.

- The backend prepares a data package containing currency names, symbols, exchange rates, and country flags, which is sent to the frontend via REST API.
- Secure interaction between the server and client is ensured by configuring HTTPS and supporting CORS.

2. Transaction History:

Django - Angular convertor

Make conversion
History access

Since date

Until date

Min amount

Max amount

From

✓

Select currency

To

✓

Select currency

Request data

Ilustración 2 History request page view

- The application includes a second tab for displaying the transaction history.
- Each operation is saved in an SQLite database.
- Users can filter transaction history by the following parameters:
 - Source currency.
 - Target currency.
 - Amount.
 - Date or date range.
- Filters generate a request to the backend (POST or GET), which is processed by Django.
- Filtered data is returned to the frontend and displayed in a user-friendly format.

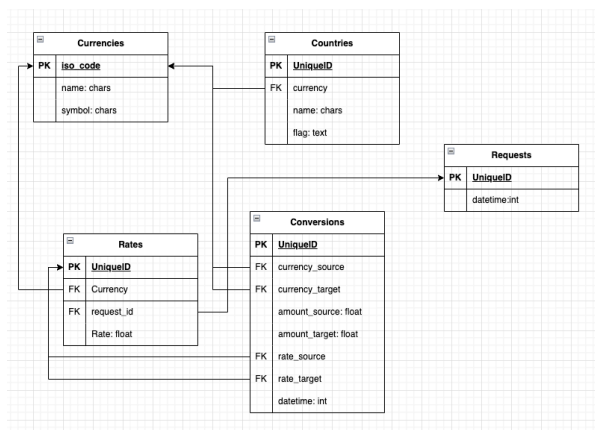


Ilustración 3 DB tables structure

3. Component Reusability:

- To unify the interface and optimize development time, an Angular metaclass was used.
 - The metaclass was applied to create components for displaying currencies in both the conversion and transaction history sections.
-

Technologies Used:

- **Frontend:** Angular, TypeScript, HTML5, CSS3
 - **Backend:** Django, SQLite
 - **Other:** REST API, HTTPS, CORS
-

What I Learned:

- Developing and configuring custom UI components in Angular.
- Organizing frontend-backend interaction via REST API.
- Configuring HTTPS and supporting CORS for secure communication.
- Reusing code through Angular metaclasses.
- Filtering data and working with an SQLite database.

GITHUB link: [Build project](#), [Angular frontend](#)