TACS 25/26 Assignment

Second Delivery (DSML Report) - group 08

Advanced Software Construction Techniques

Community Platform DSML Customization

1. Introduction

In the context of **Model-Driven Engineering (MDE)**, **Domain-Specific Modeling Languages (DSMLs)** play a fundamental role in enabling controlled variability and systematic customization without requiring modifications to the underlying system architecture or source code. Instead of embedding configuration logic directly into implementation artifacts, DSMLs allow variability to be expressed declaratively and processed automatically by model transformations.

In this project, a **DSML Customization** was designed to support the generation of a *family of community exchange applications*. Each generated application corresponds to a specific local community and reflects its particular requirements, policies, and operational constraints. This approach ensures that different communities can obtain customized applications while preserving a common architectural foundation.

2. Purpose of the DSML Customization

The Customization defines **what can vary** across different instances of the application family. Rather than developing a single monolithic application with fixed functionality, the DSML enables the generation of **multiple tailored applications** from the same core models.

Through the DSML, communities can decide, for example:

- Which payment methods are available (e.g., MB Way, Multibanco, PayPal);
- Whether ratings are simple or bidirectional;
- Whether subcommunities are supported.

By encapsulating these decisions in a DSML, the system **separates configuration concerns from implementation concerns**, improving maintainability, reuse, and scalability. This separation follows the principles of **software product-line engineering**, where a single platform supports multiple derived products through controlled variability.

3. Customization Mechanism Overview

The customization of the community exchange application is performed through a **Domain-Specific Modeling Language (DSML)** exposed to the user via a **block-based graphical editor**. This editor allows community administrators to configure the application visually, without interacting directly with code, models, or configuration files.

The configuration process starts from a **root AppConfig block**, which represents the entire application instance. This block contains global properties such as the application name, visual identity (colors and logo), and high-level options like subcommunity support. From this root block, users progressively add and configure specialized blocks that correspond to different functional areas of the application.

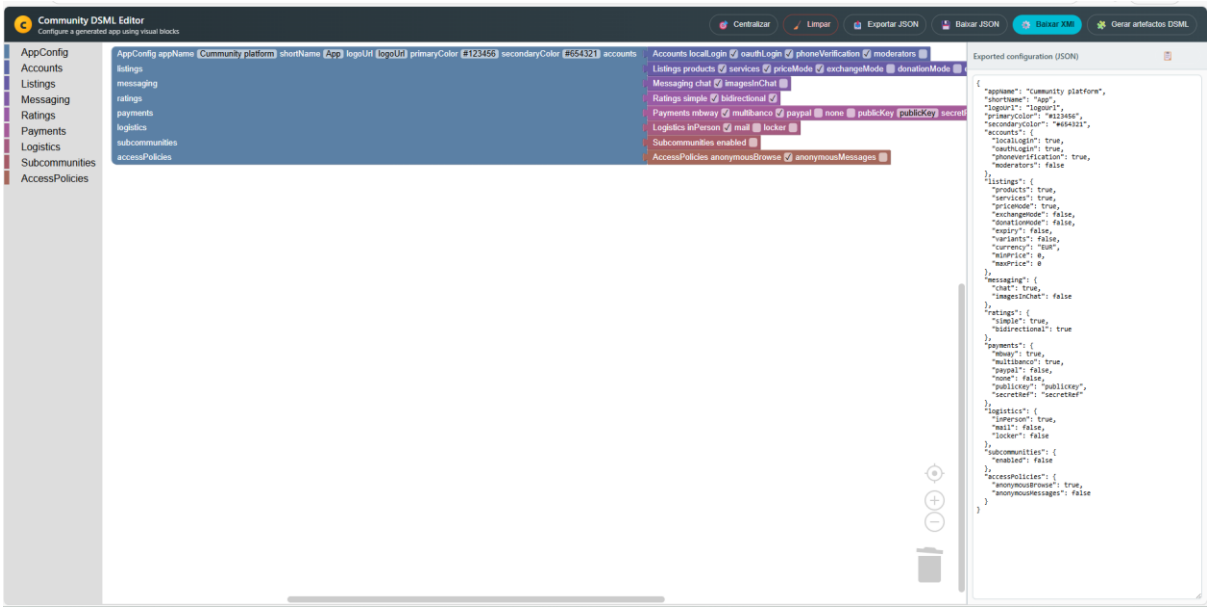
Each functional area—such as Accounts, Listings, Messaging, Ratings, Payments, Logistics, and Access Policies—is represented by a **dedicated configuration block**. These blocks expose only the options that are valid for that domain (for example, payment methods in Payments, or chat options in Messaging), guiding the user toward valid configurations.

The graphical editor enforces **structural correctness** by allowing blocks to be connected only in predefined locations. This prevents invalid configurations at the syntactic level. Additionally, the configuration values selected in the blocks are validated by **semantic constraints** defined in the DSML (e.g., payments require price mode to be enabled, only one payment method can be selected).

Once the configuration is completed, the editor can automatically export it as a **DSML instance**, either in JSON or XMI format. This exported model serves as the input for the subsequent transformation steps of the pipeline, directly influencing which features, screens, and components are generated in the final application.

Through this approach, the customization mechanism provides a **clear separation between configuration and implementation**, enabling non-technical users to tailor complex applications while preserving consistency and correctness across all generated artifacts.

Customization screen:



XMI Exported Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<custom:AppConfig xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI" xmlns:custom="http://www.example.org/custom" appName="Community platform" shortName="App" logoUrl="logoUrl" primaryColor="#123456" secondaryColor="#654321">
  <accounts localLogin="true" oauthLogin="true" phoneVerification="true" moderators="false"/>
  <listings products="true" services="true" priceMode="true" exchangeMode="false" donationMode="false" expiry="false" variants="false" currency="EUR" minPrice="0" maxPrice="0"/>
  <messaging chat="true" imagesInChat="false"/>
  <ratings simple="true" bidirectional="true"/>
  <payments mbway="true" multibanco="true" paypal="false" none="false" publicKey="publicKey" secretRef="secretRef"/>
  <logistics inPerson="true" mail="false" locker="false"/>
  <subcommunities enabled="false"/>
  <accessPolicies anonymousBrowse="true" anonymousMessages="false"/>
</custom:AppConfig>
```