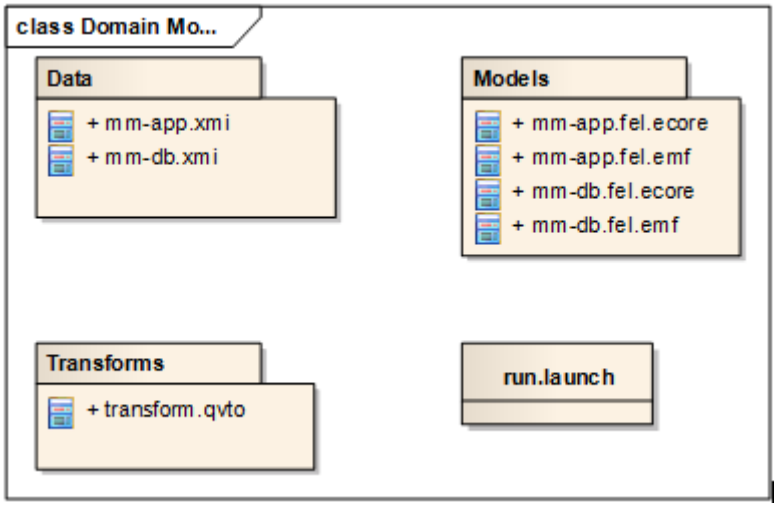
Návrh

Struktura aplikace

Na následujícím obrázku můžete vidět jednotlivé stavební části námi navrhované aplikace.



Obrázek 1 - Zde jsou znázorněny jednotlivé adresáře a soubory naší aplikace

Adresář Data

Adresář data obsahuje konkrétní modely dat. Konkrétními modely rozumíme takové modely, které odpovídají své specifikaci, neboli jejich struktura je popsaná metamodely z adresáře models. V našem případě se v tomto adresáři nacházejí dva soubory. Jedná se konkrétně o soubory mm-app.xmi a mm-db.xmi. Tyto soubory reprezentují konkrétní data, s kterými navržená transformace pracuje.

Soubor mm-app.xmi

Tento soubor je vstupní soubor pro transformace, který obsahuje konkrétní data (třídy, atributy, jednotlivé vztahy mezi třídami), kterými je struktura modelu tvořena. Nad těmito daty jsou prováděny transformace, které data přetvářejí do struktury popsané jiným metamodelem.

Soubor mm-db.xmi

Jedná se o výstupní soubor, který je vytvořen v průběhu transformací. Obsahuje data, která byla popsána v souboru mm-app.xmi, ovšem tyto data odpovídají jiné struktuře, která je popsaná příslušným metamodelem.

Adresář Models

Adresář models obsahuje metamodely, podle kterých se budou provádět jednotlivé transformace. Každý metamodel se zde nachází ve dvou formách. První formou je zápis metamodelu v jazyku Emfatic (soubor s příponou .emf), který je pro člověka velice pěkně čitelný. Druhý soubor je vygenerovaný z Emfatic souboru a reprezentuje zápis v jazyku Ecore (přípona .ecore). Ecore metamodel je použit jako vstupní i výstupní metamodel pro transformace. Je tomu především proto,

že zápis v Ecore je dobře strukturovaný a tudíž pársovatelný počítačem. Metamodely udávají strukturu, kterou konkrétní modely musí splňovat.

Soubory mm-app.fel.ecore a mm.app.fel.emf

V těchto souborech se nachází metamodely, podle kterých se budou jednotlivé transformace provádět. Metamodely vlastně předepisují konstrukty a omezení, které budou muset konkrétní modely splňovat. V těchto metamodelech se předepisuje struktura pro aplikační modely (vstupní modely do transformace). Definují se zde třídy, atributy, operace.

Soubory mm-db.fel.ecore a mm.db.fel.emf

Tyto soubory popisují metamodel pro výstupní data z transformace. Tento metamodel se zabývá definicí modelů na úrovni databáze. Je zde specifikováno, jak má vypadat tabulka, sloupec, primary key, forein key, indexy, sekvence, constrains a jsou zde také popsány základní DDL operace, jako vytvoř tabulku, přejmenuj sloupec atd.

Adresář Transforms

Tento adresář obsahuje soubory, které jsou zodpovědné za samotnou transformaci. V našem případě se zde nachází pouze jeden soubor a to transform.qvto

Soubor transform.gvto

Tento soubor se zabývá konkrétní transformací vstupního modelu na výstupní. V našem případě má na vstupu model, který odpovídá metamodelu mm-app.fel.ecore, jehož data převádí na výstupní model, který odpovídá modelu mm-db.fel.ecore. Jinými slovy, převádí strukturu, která je popsaná pomocí tříd, atributů, atd. na strukturu, která odpovídá databázovému světu, tedy tabulky, sloupce atd.

Soubor run.launch

Tento soubor spouští celou transformaci. Jsou zde například uvedené informace o umístění transformačních skriptů, vstupních i výstupních modelů a spoustu dalších inicializací.

Technologie

Použité technologie do značné míry zastřešuje EMF, **neboli Eclipse Modeling Framework**. Jedná se o souhrn modelovacích nástrojů pro vývojové prostředí Eclipse. Eclipse i celý koncept EMF jsme použili z důvodu kompatibility s aplikací, do které bude náš modul umístěn.

Používali jsme především jazyky, které popisují modely. Jedná se zejména o **Ecore**, který je založen na XML, a **Emfatic**, který je velice přehledný a svým zápisem se podobá programovacímu jazyku Java. Mezi těmito modely se dá jednoduše přecházet (z jednoho zápisu se dá vygenerovat zápis druhý). V jazyku Emfatic jsme modely vytvářeli, následně jsme vygenerovali jemu odpovídající model v Ecore a ten jsme dále programově zpracovávali.

Dále jsme používali transformační jazyk. Naše volba padla na transformační jazyk **QVT Operational**. Tento jazyk jsme zvolili kvůli jeho snadné integraci do Eclipse, ale především proto, že zadavatel má s tímto jazykem bohaté zkušenosti. Jazyk QVT nám umožní převést datovou strukturu odpovídající modelu A (původní) na strukturu odpovídající modelu B (nový model) pomocí definované transformace.

Upozornění na budoucí místa interakce/interface

EMF (Emfatic)

Ve formátu EMF nám bude poskytnuta struktura aplikace a sada změn, které se objevily při přechodu na novou verzi.

Postgres SQL

Relační databáze, nad kterou budou prováděny změny v modelu.