Uma DSL para gerar APIs RESTFul em Haskell

Apresentação sobre uma Domain Specific Language inovadora para automatizar a criação do backend de aplicações web em Haskell.

Os integrantes:

Angemydelson Saint-Bert,
Sebastien Lionel Lubin,
Kerby Lovince

CIÊNCIA DA COMPUTAÇÃO UFRJ

Autores do artigo

Gr. Nataniel Fernando Lemes Siebeneichle

Dr. Felipe Grando

Dr. Giancarlo Dondoni Salton

Dr. Samuel da Silva Feitosa

Introdução

Este artigo aborda a necessidade de simplificar o desenvolvimento de aplicações web em Haskell. Propomos uma DSL inovadora que automatiza a criação do backend de aplicações web por meio de uma API RESTful de alto nível. Essa abordagem elimina a tediosa tarefa de desenvolver funcionalidades CRUD, permitindo que os desenvolvedores se concentrem em áreas mais estratégicas. Todo o código-fonte deste projeto foi desenvolvido em Haskell (versão 9.2.7), com a biblioteca Servant (versão 0.19.1).

Fundamentação Teórica

1 Model-driven Engineering

Model-Driven Engineering (MDE) é uma abordagem na qual siste-

mas de software são modelados em um nível abstrato,

2 Linguagens de Domínio Específico (DSL)

As Linguagens de Domínio Específico (DSL) são linguagens desenvolvidas e adaptadas para atender a um domínio específico de aplicação [12].

3 REST

Web Services REST (REpresentational State Transfer) oferecem

uma interface para CRUD [14], porém, não somente a essas operações.

Fundamentação Teórica

4 CRUD

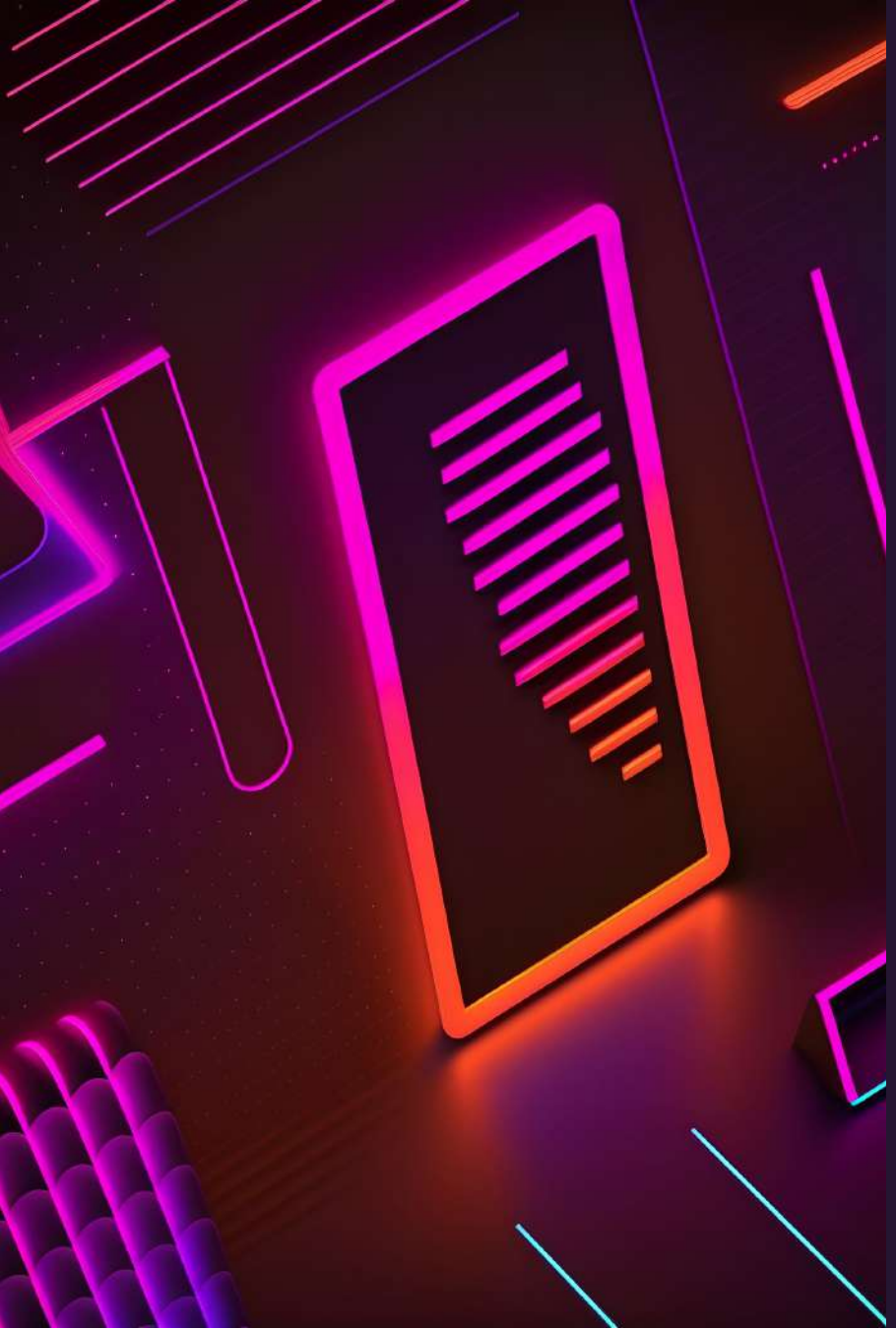
As operações de CRUD (Create, Retrieve, Update e Delete) são funcionalidades essenciais que são implementadas em conjunto com um banco de dados

5 Template Haskell

É uma extensão para o Haskell que permite fazer meta-programação em tempo de compilação.

Uma DSL para gerar APIs REST

Este artigo descreve uma Domain Specific Language (DSL) baseada em arquivos JSON para definir dados e funcionalidades de uma API web. A sintaxe e as palavras reservadas são especificadas em um arquivo-fonte que contém todas as informações necessárias para a geração de código para o backend de uma API web.



Uma DSL para gerar APIs RESTFul em Haskell

A especificação de um determinado CRUD por meio da DSL proposta se dá a partir da criação de um arquivo-fonte, o qual deve conter todas as informações necessárias para o processo de geração de código para o backend de uma API Web.

Ferramentas utilizadas

As principais ferramentas utilizadas para o desenvolvimento deste projeto fazem parte do ecossistema da linguagem Haskell. Dentre elas, destacam-se o framework Servant, a biblioteca Persistent, e o próprio Template Haskell. Todas essas ferramentas são comumente adotadas durante o desenvolvimento em Haskell.

Interpretação da DSL

Apresentamos o motor do framework proposto, discorrendo sobre como a DSL é interpretada para a criação e manipulação dos modelos de dados, além da criação dos endpoints para acesso aos Web services.

A API está organizada da seguinte forma:

- Definição de um CRUD através da DSL
- Geração dos Modelos de Dados
- Geração das Rotas e Handlers
- Implementação de Ações Customizadas
- Processador Genérico para Operações CRUD

Trabalhos Relacionados

1 Gómez et al.

Descrevem em seu artigo uma linguagem dedomínio específico chamada CRUDyLeaf, que é usada para gerar APIs RESTful usando o framework Spring Boot.

2 Livraghi

Apresenta um sistema para geração automática de aplicações CRUD.

3 Rodriguez-Echeverria et al.

Propõe uma abordagem para gerar CRUDs usando uma Linguagem de Modelagem de Fluxo de Interação (IFML) desenvolvida como um plug-in do Eclipse na linguagem Java.

Conclusão

Introduzimos uma DSL para a geração automatizada de APIs RESTful em Haskell. A utilização de DSLs em conjunto com Haskell simplifica o desenvolvimento de APIs, permitindo uma codificação mais rápida e econômica. Nossa ferramenta automatiza tarefas repetitivas, permitindo que os desenvolvedores se concentrem em aspectos mais cruciais do projeto. Os CRUDs podem ser automatizados completamente, enquanto as partes específicas e complexas do sistema podem ser desenvolvidas utilizando a abordagem tradicional de Haskell.

Referências

1. Michael D Adams and Thomas M DuBuisson. 2012. Template your boilerplate: Using Template Haskell for efficient generic programming. ACM SIGPLAN Notices 47, 12 (2012), 13–24.
2. James Cheney and Ralf Hinze. 2003. Phantom types. Technical Report. Technical Report CUCIS TR20003-1901, Cornell University.
3. Marco Antonio Copetti et al. 2012. Um processo integrado para qualidade em model-driven engineering. (2012).
4. Antonio Delgado, A Estepa, and Rafael Estepa. 2007. WAINE-Automatic Generator of Web Based Applications. In International Conference on Web Information Systems and Technologies, Vol. 2. SCITEPRESS, 226–233.
5. Hamza Ed-Douibi, Javier Luis Cánovas Izquierdo, Abel Gómez, Massimo Tisi, and Jordi Cabot. 2016. EMF-REST: generation of RESTful APIs from models. In Proceedings of the 31st Annual ACM Symposium on Applied Computing. 1446–1453.