Зборник радова Факултета техничких наука, Нови Сад

УДК: 4.41
ДОИ: https://doi.org/10.24867/35BE27Jovic

Генератор кода за имплементацију иницијалне верзије микросервисне апликације

Code Generator for Implementing the Initial Version of a Microservice Application

Никола Јовић, Факултет техничких наука, Нови Сад

Студијски програм – СОФТВЕРСКО ИНЖЕЊЕРСТВО И ИНФОРМАЦИОНЕ ТЕХНОЛОГИЈЕ

Кратак садржај – *У овом раду представљен је алат за аутоматско генерисање иницијалне верзије микросервисне апликације на основу YAML спецификације. Рад обухвата анализу постојећих решења у области аутоматизованог генерисања микросервиса, као и опис архитектуре и имплементације алата реализованог као плагин за IntelliJ IDEA развојно окружење. Приказане карактеристике и функционалности алата указују на његову применљивост у иницијалној фази развоја микросервисних апликација.*

Кључне речи: *генератор кода, микросервисна апликација, YAML*

Abstract – *This paper presents a tool for automated generation of the initial version of a microservice application based on the YAML specification. The paper includes an analysis of existing solutions in the field of automated microservice generation, as well as a description of the architecture and implementation of the tool developed as a plugin for the IntelliJ IDEA development environment. The presented features and functionalities of the tool indicate its applicability in the initial phase of microservice application development.*

Keywords: *code generator, microservice application, YAML*

НАПОМЕНА: Овај рад проистекао је из мастер рада чији ментор је била др Гордана Милосављевић, редовни професор.

1. УВОД

Развој микросервисних апликација представља један од кључних приступа у савременом софтверском инжењерству који омогућава независан развој појединачних компоненти система, бољу скалабилност и једноставније одржавање [1]. Креирање микросервисне апликације у почетној фази развоја често подразумева велики број понављајућих корака, као што су подешавање конфигурације пројекта и имплементација основних слојева апликације. Ови задаци могу значајно успорити развој и повећати могућност настанка грешака услед ручног писања кода.

Наведени проблеми се превазилазе употребом генератора кода који омогућавају убрзавање развоја и унапређење стандардизације кода. Узимајући у обзир наведене изазове, циљ овог рада је развијање новог алата који би пружао аутоматизовано генерисање почетне микросервисне апликације и тиме програмерима омогућио бржи, једноставнији и ефикаснији процес развоја микросервисних апликација.

2. ПРИКАЗ СТАЊА У ОБЛАСТИ

Алати за генерисање кода и аутоматизован развој представљају иновативан корак ка унапређењу процеса развоја софтвера. У оквиру академских истраживања развијени су алати засновани на формалним моделима, као што су *FSMicroGenerator* и *Zynerator*. *FSMicroGenerator* користи модел коначних аутомата за формално описивање понашања система, након чега се из модела генерише изворни код микросервисне апликације [2]. Овај приступ обезбеђује јасну структуру и доследност, али је ограничен у погледу проширивости и применљивости на комплексније системе. *Zynerator* је заснован на архитектури вођеној моделом и омогућава трансформацију апстрактних доменских модела у изворни код [3]. Он обезбеђује добру поновну употребу и стандардизацију, али његова примена захтева додатно познавање моделских концепата.

У индустрији постоје решења која програмерима нуде практичне платформе са високим степеном аутоматизације, као што су *JHipster* и *Micronaut Launch*. *JHipster* омогућава генерисање комплетне микросервисне апликације [4], али уз велику сложеност и зависност од великог броја технологија. *Micronaut Launch* пружа брзо генерисање појединачних сервиса који прате праксе *Micronaut* фрејмворка и имају добре перформансе [5], али не подржава централизовано генерисање потпуне микросервисне апликације.

Анализа постојећих решења указује на потребу за новим алатом који ће комбиновати једноставност употребе, практичну применљивост и лаку интеграцију у развојно окружење, уз могућност генерисања микросервисне апликације на основу претходно дефинисане спецификације. Овим би се

убрзао развој, смањио понављајући рад и могућност појаве грешака.

3. ПРОЈЕКТОВАЊЕ АЛАТА *MICROGENERATOR*

Пројектовани алат треба да обезбеди брзо, једноставно и поуздано генерисање почетне верзије микросервисне апликације на основу формално дефинисаног модела. Основу рада алата представља декларативна *YAML* спецификација. У оквиру ње се дефинишу основни метаподаци микросервиса, конфигурациони параметри, зависности, подешавања базе података и доменски модели.

3.1. Функционалности алата

Алат омогућава избор и учитавање *YAML* датотеке, врши валидацију њене структуре и садржаја, као и аутоматско генерисање *Spring Boot* пројекта за сваки дефинисани микросервис. На основу описаних доменских модела алат генерише основне компоненте апликације, укључујући ентитете, репозиторијуме, сервисе и *REST* контролере са ендпоинтима за креирање, читање, измену и брисање података.

Поред функционалних, алат мора испуњавати и нефункционалне захтеве који обухватају: да је имплементиран као *IntelliJ IDEA* плагин, да је структура креираних датотека и класа стандардизована и усклађена са праксама *Spring Boot* екосистема и да је дизајниран тако да се у будућности може надоградити и унапредити новим шаблонима и функционалностима.

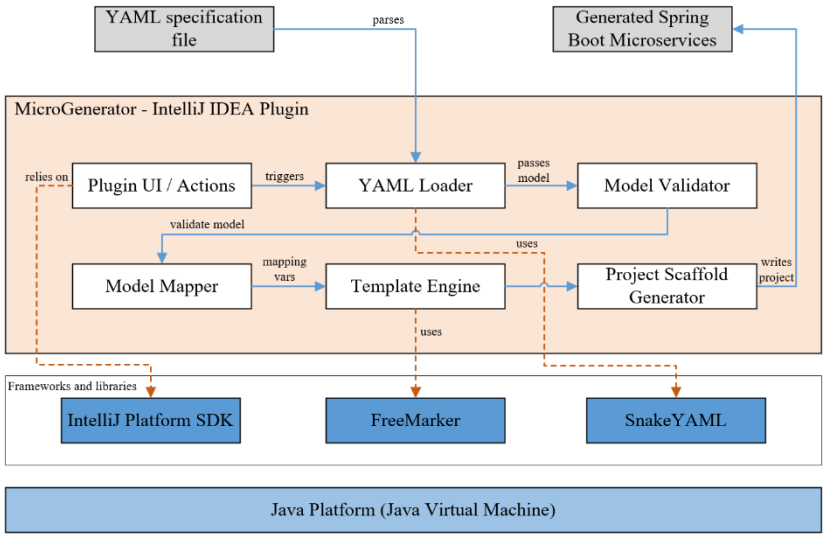
3.2. Архитектура алата

Дизајн архитектуре алата заснован је на принципима слојевите организације, одвајања одговорности и јасној подели улога за сваку компоненту у систему. Централни део архитектуре чине модули који реализују ток обраде, од корисничког интерфејса, преко учитавања и валидације *YAML* спецификације, до трансформације у интерни модел и генерисања изворног кода.

Кориснички интерфејс представља улазну тачку за покретање процеса генерисања. Он прикупља параметре и прослеђује их одговарајућем модулу на даљу обраду. Оваква интеграција у развојно окружење елиминише потребу за инталацијом посебног екстерног алата. Након избора *YAML* датотеке, активира се модул задужен за учитавање спецификације који парсира садржај и трансформише га у интерну репрезентацију. У случају детектованих грешака кориснику се приказује порука да је потребно исправити улазну спецификацију.

Циљ модула за трансформацију је претварање садржаја интерне репрезентације у структуре података које представљају домен микросервиса. Генерисање изворног кода се реализује коришћењем механизма шаблона. Шаблони дефинишу структуру свих делова микросервиса: модела, репозиторијума, сервисног слоја, контролера и конфигурација.

Модул за генерисање кода, након што се шаблони попуне подацима, добијени текстуални садржај записује у датотеке на диску. Ова компонента је одговорна за креирање целокупне структуре директоријума *Spring Boot* пројекта, креирање *Java* класа у одговарајућим пакетима, формирање *Maven* конфигурације и креирање конфигурационих датотека.



Слика 1. Архитектура алата *MicroGenerator*

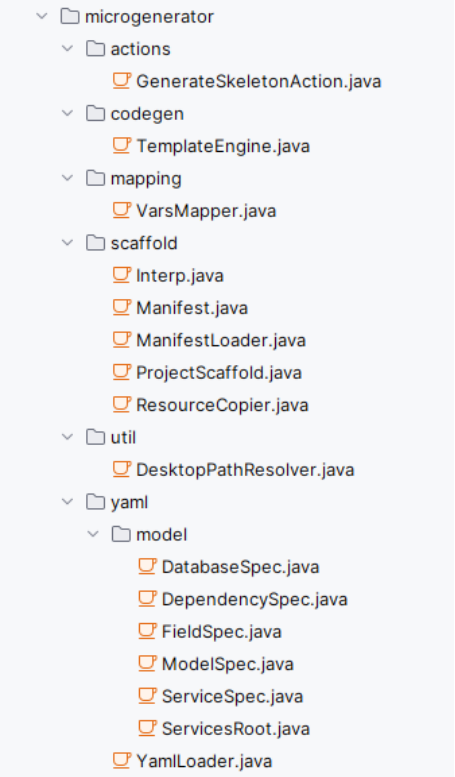
4. ИМПЛЕМЕНТАЦИЈА

Имплементација предложеног решења реализована је кроз развој алата у виду плагина за развојно окружење *IntelliJ IDEA*. Овакав приступ омогућава директну интеграцију у окружење које програмери већ користе, чиме се алат природно уклапа у постојећи радни процес, омогућава његова примена без нарушавања уобичајеног тока развоја и избегава потреба за инсталацијом додатног софтвера.

4.1. Структура пројекта и имплементација модула

Структура пројекта организована је модуларно, са јасном поделом одговорности између појединих компоненти. Улазну тачку представља корисничка акција која се покреће из развојног окружења. Ова

компонента преузима контекст активног пројекта, иницира избор *YAML* спецификације и покреће даљи ток обраде, и на тај начин је интеракција корисника са алатом сведена на мали број једноставних корака.



Слика 2. Структура пакета алата *MicroGenerator*

Учитавање и парсирање спецификације реализовано је у оквиру посебног модула задуженог за рад са улазним моделом. За обраду *YAML* формата коришћена је библиотека *SnakeYAML*, која омогућава директно пресликавање структуре спецификације у типизиране *Java* објекте [6]. Оваквим приступом обезбеђена је типска сигурност и поједностављена валидација улазних података. Основне провере, постојање обавезних секција и исправност структуре, извршавају се приликом самог учитавања спецификације чиме се спречава рад са неконзистентним улазом.

Након успешног учитавања, интерни модел се трансформише у структуру података погодну за механизам шаблона. Овај корак је реализован у посебном слоју за мапирање који има улогу да из улазне спецификације преузме све параметре неопходне за генерисање кода. Тиме је постигнуто раздвајање логике интерпретације од логике шаблона, чиме постигнута је боља одрживост алата.

Генерисање изворног кода засновано је на употреби *FreeMarker* шаблона, који дефинишу структуру генерисаних *Java* класа и конфигурационих датотека [7], док се конкретне вредности уносе на основу параметара припремљених у претходној фази. Шаблони су организовани тако да постоје они који се генеришу за сваки доменски модел појединачно и они који се генеришу једном по микросервису. Део датотека се не генерише из шаблона, већ се копирају као статички ресурси у пројекат.

Креирање физичке структуре микросервисне апликације реализовано је у посебном модулу који је задужен за оркестрацију процеса генерисања директоријума и уписа датотека на диск. Структура пројекта је дефинисана у конфигурационом манифесту, који описује које се датотеке генеришу, које се копирају и где се смештају у оквиру пројекта. Овакав приступ омогућава да генерисана микросервисна апликација буде у потпуности усклађена са *Maven* и *Spring Boot* конвенцијама и да је спремна за покретање.

4.2. *YAML* спецификација

YAML спецификација представља централни улазни артефакт алата и служи за декларативни опис микросервисне апликације. Спецификација омогућава дефинисање једног или више микросервиса, њихових зависности, конфигурације базе података и доменских модела. Доменски модели се описују именима, називима табела и поља, на основу чега се аутоматски генеришу одговарајући *JPA* ентитети. Подржано је и моделовање релација између ентитета, који се током генерисања пресликавају на одговарајуће *JPA* асоцијације.

4.3. Изазови током имплементације

Током имплементације уочено је више техничких и концептуалних изазова. Један од кључних је био дефинисање *YAML* спецификације која је довољно флексибилна да опише различите микросервисе, а истовремено довољно једноставна за коришћење. То је превазиђено увођењем јасно дефинисане хијерархијске структуре и типизираних моделских класа.

Важан имплементациони изазов је био избегавање уношења пословне логике у шаблоне за генерисање кода. Уколико би шаблони садржали сложене услове и трансформације, њихово одржавање и проширивање би било отежано. Проблем је решен увођењем засебног слоја за мапирање података, који припрема све неопходне параметре пре фазе генерисања.

Обезбеђивање доследне и исправне структуре генерисане микросервисне апликације је постигнуто увођењем конфигурационог манифеста који дефинише структуру пројекта, распоређивање датотека и начин њиховог креирања.

5. АНАЛИЗА РЕЗУЛТАТА

Анализа резултата развоја алата *MicroGenerator* усмерена је на процену његових квалитативних атрибута и поређења са постојећим решењима за аутоматизовано генерисање микросервисних апликација.

5.1. Квалитативни атрибути алата *MicroGenerator*

Један од кључних квалитативних атрибута алата је употребљивост, која произилази из његове директне интеграције у развојно окружење. Корисник може да покрене процес генерисања без напуштања окружења у коме већ ради што поједностављује радни ток и смањује потребу за додатним алатима. Употреба *YAML* спецификације као улазног формата омогућава декларативан опис микросервисне архитектуре, чиме се смањује сложеност конфигурационих корака.

Проширивост алата остварена је кроз модуларну архитектуру и јасну поделу одговорности између компоненти за обраду спецификације, мапирање података и генерисање кода.
Оваква организација омогућава додавање нових шаблона, увођење подршке за додатне технологије или проширење улазне спецификације без значајних измена постојећег кода.
Одрживост решења обезбеђена је кроз модуларну организацију и коришћење шаблона за генерисање кода. Измене у начину рада алата могу се реализовати без негативног утицаја на остале компоненте система.
Перформансе алата посматрају се у односну на његову улогу у иницијалној фази развоја микросервисних апликација, при чему је процес учитавања спецификације и генерисања кода довољно ефикасан и не утиче негативно на стабилност развојног окружења.

5.2. Поређење са постојећим алатима

У поређењу са постојећим алатима, *MicroGenerator* заузима специфичну позицију између академских прототипова и индустријских решења. За разлику од алата заснованих на формалним моделима, као што су *FSMicroGenerator* и *Zynerator*, предложено решење не захтева употребу специјализованих моделских језика, већ се ослања на *YAML* као широко прихваћен формат.
У односу на комплексна индустријска решења, попут *JHipster*-а, *MicroGenerator* нуди једноставнији приступ усмерен на аутоматизацију почетне фазе развоја, без увођења великог броја конфигурационих корака. За разлику од *Micronaut Launch*-а који омогућава генерисање појединачних сервиса, *MicroGenerator* подржава централизовано генерисање комплетне микросервисне апликације.
MicroGenerator није директна замена за постојеће алате, али његово коришћење декларативног описа микросервисне архитектуре и интеграција у развојно окружење издвајају га као алат за убрзање почетне фазе развоја микросервисних апликација.

6. ЗАКЉУЧАК

У овом раду представљен је процес пројектовања, имплементације и евалуације алата *MicroGenerator*, намењеног аутоматизованом генерисању иницијалне верзије микросервисне апликације на основу *YAML* спецификације.
Кроз анализу постојећих академских и индустријских решења уочена су ограничења која се односе на сложеност употребе, потребу за формалним моделима или недостатак централизованог приступа генерисању микросервисних апликација. На основу тога развијен је алат који комбинује декларативни приступ опису архитектуре, једноставност употребе и директну интеграцију у развојно окружење *IntelliJ IDEA*. Имплементација алата као плагина омогућила је практичну примену у реалним развојним сценаријима.
Анализа резултата показује да *MicroGenerator* успешно испуњава постављене циљеве, јер смањује понављајући рад, повећава стандардизацију кода и убрзава развој у почетној фази изградње микросервисних апликација. Представљено решење пружа основу за даља унапређења, што може бити проширење улазне спецификације или подршка за напредне инфрастуктурне компоненте, чиме се отвара простор за будући развој алата и његову примену.

7. ЛИТЕРАТУРА

[1] N. Dragoni, S. Giallorenzo, A. Lluch Lafuente, M. Mazzara, F. Montesi, R. Mustafin, and L. Safina, “Microservices: Yesterday, Today, and Tomorrow,” Present and Ulterior Software Engineering, Springer, 2017, pp. 195-216. doi: 10.1007/978-3-319-67425-4_12.

[2] S. Khalfaoui, H. Khalfaoui, and A. Azmani, “Microservices-Driven Automation in Full-Stack Development: Bridging Efficiency and Innovation With FSMicroGenerator,” IEEE Access, vol. 13, pp. 125131-125156, 2025. doi: 10.1109/ACCESS.2025.3589285.

[3] Y. Zouani, and M. Lachgar, “Zynerator: Bridging Model-Driven Architecture and Microservices for Enhanced Software Development,” Electronics, vol. 13, no. 12, art. 2237, 2024. doi: 10.3390/electronics13122237.

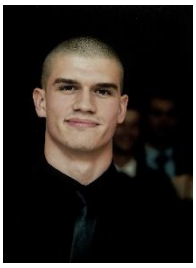
[4] JHipster, “JHipster Official Website.” Available: https://www.jhipster.tech (accessed: Oct. 10, 2025).

[5] Micronaut, “Micronaut Launch.” Available: https://micronaut.io/launch (accessed: Oct. 10, 2025).

[6] SnakeYAML, “SnakeYAML Project.” Available: https://bitbucket.org/snakeyaml/snakeyaml (accessed: Nov. 30, 2025).

[7] Apache, “FreeMarker.” Available: https://freemarker.apache.org/index.html (accessed: Nov. 30, 2025).

Кратка биографија:



Никола Јовић рођен је у Лозници 1999. године. Основне академске студије завршио је на Војној академији у Београду. Мастер рад на Факултету техничких наука у Новом Саду, смер Софтверско инжењерство и информационе технологије, одбранио је 2026. године.

Контакт:jovnikola99@gmail.com