



УНИВЕРЗИТЕТ У НОВОМ САДУ  
ФАКУЛТЕТ ТЕХНИЧКИХ  
НАУКА У НОВОМ САДУ

---



Никола Трајковић

**Дизајн и реализација  
дистрибуираног *CI/CD* алата са  
подршком за *2FA*,  
нотификације и корисничку  
аналитику**

ЗАВРШНИ РАД

Основне академске студије

Нови Сад, 2025



	УНИВЕРЗИТЕТ У НОВОМ САДУ • ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА 21000 НОВИ САД, Трг Доситеја Обрадовића 6	Број:
	<b>ЗАДАТАК ЗА ЗАВРШНИ РАД</b>	Датум:

(Податке уноси предметни наставник - ментор)

Студијски програм:	Софтверско инжењерство и информационе технологије		
Студент:	Никола Трајковић	Број индекса:	SV45/2021
Степен и врста студија:	Основне академске студије		
Област:	Софтверско инжењерство и информационе технологије		
Ментор:	Бранко Милосављевић		
НА ОСНОВУ ПОДНЕТЕ ПРИЈАВЕ, ПРИЛОЖЕНЕ ДОКУМЕНТАЦИЈЕ И ОДРЕДБИ СТАТУТА ФАКУЛТЕТА ИЗДАЈЕ СЕ ЗАДАТАК ЗА ЗАВРШНИ РАД, СА СЛЕДЕЋИМ ЕЛЕМЕНТИМА: <ul style="list-style-type: none"> <li>- проблем – тема рада;</li> <li>- начин решавања проблема и начин практичне провере резултата рада, ако је таква провера неопходна;</li> </ul>			

### НАСЛОВ ЗАВРШНОГ РАДА:

Дизајн и реализација дистрибуираног <i>CI/CD</i> алата са подршком за <i>2FA</i> , нотификације и корисничку аналитику
--

### ТЕКСТ ЗАДАТКА:

Анализирати концепте, архитектуру и начин рада постојећих <i>CI/CD</i> алата. Анализирати технологије за имплементацију двофакторске аутентификације. Анализирати механизме за интеграцију <i>CI/CD</i> система са системима за контролу верзија. Дефинисати архитектуру и имплементирати <i>CI/CD</i> систем који омогућава извршавање задатака на више чворова помоћу дистрибуираних агената, поседује двофакторску аутентификацију, нотификације вођене догађајима и кориснички интерфејс за надзор рада система. Документовати решење и дискутовати добијене резултате.
---

Руководилац студијског програма:	Ментор рада:

Примерак за: <input type="checkbox"/> - Студента; <input type="checkbox"/> - Ментора
--





УНИВЕРЗИТЕТ У НОВОМ САДУ • ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА  
21000 НОВИ САД, Трг Доситеја Обрадовића 6

## КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

Редни број, **РБР**:

Идентификациони број, **ИБР**:

Тип документације, **ТД**:

Монографска документација

Тип записа, **ТЗ**:

Текстуални штампани материјал

Врста рада, **ВР**:

Дипломски - бечелор рад

Аутор, **АУ**:

Никола Трајковић

Ментор, **МН**:

Др Бранко Милосављевић, редовни професор

Наслов рада, **НР**:

Дизајн и реализација дистрибуираног *CI/CD* алата са подршком за *2FA*,  
нотификације и корисничку аналитику

Језик публикације, **ЈП**:

српски/ћирилица

Језик извода, **ЈИ**:

српски/енглески

Земља публиковања, **ЗП**:

Република Србија

Уже географско подручје, **УГП**:

Војводина

Година, **ГО**:

2025

Издавач, **ИЗ**:

Ауторски репринт

Место и адреса, **МА**:

Нови сад, трг Доситеја Обрадовића 6

Физички опис рада, **ФО**:

(поглавља/страница/ цитата/табела/слика/графика/прилога)

7/25/4/0/3/0/0

Научна област, **НО**:

Софтверско инжењерство и  
информационе технологије

Научна дисциплина, **НД**:

Примењене рачунарске науке и информатика

Предметна одредница/Кључне речи, **ПО**:

Шаблон, завршни рад, упутство

### УДК

Чува се, **ЧУ**:

У библиотеци Факултета техничких наука, Нови Сад

Важна напомена, **ВН**:

Извод, **ИЗ**:

Овај документ представља упутство за писање завршних радова на  
Факултету техничких наука Универзитета у Новом Саду. У исто време је и  
шаблон за *Thurst*.

Датум прихватања теме, **ДП**:

Датум одбране, **ДО**:


Чланови комисије, **КО**:

Председник: Др Горан Сладић, редовни професор

Члан: Др Мирослав Зарић, редовни професор

Члан, ментор: Др Бранко Милосављевић, редовни професор

Потпис ментора

	UNIVERSITY OF NOVI SAD • FACULTY OF TECHNICAL SCIENCES 21000 NOVI SAD, Trg Dositeja Obradovića 6
	KEY WORDS DOCUMENTATION

Accession number, <b>ANO</b> :	
Identification number, <b>INO</b> :	
Document type, <b>DT</b> :	Monographic publication
Type of record, <b>TR</b> :	Textual printed material
Contents code, <b>CC</b> :	
Author, <b>AU</b> :	Nikola Trajković
Mentor, <b>MN</b> :	Branko Milosavljević, Phd., full professor
Title, <b>TI</b> :	Design and Implementation of a Distributed CI/CD Tool with Support for 2FA, Notifications, and User Analytics
Language of text, <b>LT</b> :	Serbian
Language of abstract, <b>LA</b> :	Serbian
Country of publication, <b>CP</b> :	Republic of Serbia
Locality of publication, <b>LP</b> :	Vojvodina
Publication year, <b>PY</b> :	2025
Publisher, <b>PB</b> :	Author's reprint
Publication place, <b>PP</b> :	Novi Sad, Dositeja Obradovica sq. 6
Physical description, <b>PD</b> : (chapters/pages/ref./tables/pictures/graphs/appendixes)	7/25/4/0/3/0/0
Scientific field, <b>SF</b> :	Software Engineering and Information Technologies
Scientific discipline, <b>SD</b> :	Applied computer science and informatics
Subject/Key words, <b>S/KW</b> :	Template, thesis, tutorial
<b>UC</b>	
Holding data, <b>HD</b> :	The Library of Faculty of Technical Sciences, Novi Sad, Serbia
Note, <b>N</b> :	
Abstract, <b>AB</b> :	This document provides guidelines for writing final theses at the Faculty of Technical Sciences, University of Novi Sad. At the same time, it serves as a Typst template.

Accepted by the Scientific Board on, <b>ASB</b> :				
Defended on, <b>DE</b> :				
Defended Board, <b>DB</b> :	President:	Goran Sladić, Phd., full professor		
	Member:	Miroslav Zarić, Phd., full professor		
	Member, Mentor:	Branko Milosavljević, Phd., full professor		
		<table><tr><td>Menthor's sign</td></tr><tr><td></td></tr></table>	Menthor's sign	
Menthor's sign				



УНИВЕРЗИТЕТ У НОВОМ САДУ • ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА  
21000 НОВИ САД, Трг Доситеја Обрадовића 6

## ИЗЈАВА О НЕПОСТОЈАЊУ СУКОБА ИНТЕРЕСА

Изјављујем да нисам у сукобу интереса у односу ментор – кандидат и да нисам члан породице (супружник или ванбрачни партнер, родитељ или усвојитељ, дете или усвојеник), повезано лице (крвни сродник ментора/кандидата у правој линији, односно у побочној линији закључно са другим степеном сродства, као ни физичко лице које се према другим основама и околностима може оправдано сматрати интересно повезаним са ментором или кандидатом), односно да нисам зависан/на од ментора/кандидата, да не постоје околности које би могле да утичу на моју непристрасност, нити да стичем било какве користи или погодности за себе или друго лице било позитивним или негативним исходом, као и да немам приватни интерес који утиче, може да утиче или изгледа као да утиче на однос ментор-кандидат.

У Новом Саду, дана \_\_\_\_\_

Ментор

\_\_\_\_\_

Кандидат

\_\_\_\_\_





---

# Садржај

---

1	Увод .....	1
2	CI/CD алати .....	3
2.1	<i>Jenkins</i> .....	3
2.2	<i>GitHub Actions</i> .....	3
2.3	<i>GitLab CI/CD</i> .....	4
3	Спецификација система .....	5
3.1	Функционални захтеви .....	5
3.2	Нефункционални захтеви .....	7
4	Имплементација система .....	9
4.1	Архитектура система .....	9
4.2	Безбедносни механизми .....	14
4.3	Управљање корисницима и агентима .....	14
4.4	Управљање пословима (Job-овима) .....	14
4.5	Систем нотификација .....	14
4.6	Интеграције са спољним системима .....	14
4.7	Корисничка аналитика .....	14
5	Закључак .....	15
	Биографија .....	23
	Литература .....	25

---

# Глава 1

---

## Увод

---

У индустријама као што су авијација, аутомобилска индустрија и телекомуникације, квалитет и поузданост система имају велики значај. Грешке у овим областима могу бити веома скупе или критичне. Због тога стандарди захтевају јасан преглед функционалних и нефункционалних захтева и њихову потпуну покривеност тестовима. Таква пракса омогућава већу безбедност и поузданост софтвера.

Континуирана интеграција и испорука (CI/CD) су темељ савременог софтверског инжењерства. Оне омогућавају бржу и поузданију испоруку апликација. Ипак, често не постоји јасна веза између захтева система и тестних случајева. То доводи до смањене покривености и повећава ризик од пропуштања критичних сценарија. Потребни су алати који повезују управљање захтевима и тестирање. На тај начин сви подаци се налазе на једном месту и избегава се дуплирање. Комуникација између тимова је боља, а захтеви и тестови су лако повезани и прегледни.

Овај рад описује развој CI/CD алата заснованог на мастер-агент архитектури. Алат је намењен сложеним и безбедносно осетљивим окружењима. Омогућава централизовано управљање и координацију аутоматизованих радних токова у дистрибуираном систему. Тако се постижу већа скалабилност и флексибилност. Подржани су кораци као што су клонирање складишта, пренос датотека, извршавање наредби и генерисање извештаја. Безбедност је обезбеђена двофакторском аутентикацијом (2FA) и шифрованом комуникацијом између чворова. Алат садржи аналитике за праћење перформанси и нотификације преко *Microsoft Teams*-а, *Slack*-а и саме апликације.

Циљ рада је да прикаже архитектуру, безбедносне механизме и аналитичке могућности развијеног алата. Посебна пажња посвећена је практичној примени у организацијама које захтевају високу поузданост и контролу процеса. Рад показује да савремен и интегрисан приступ аутоматизацији може унапредити развој и испоруку софтвера, чинећи их ефикаснијим, прегледнијим и лакшим за одржавање.

---

# Глава 2

---

## CI/CD алати

---

Развој софтвера данас захтева брзу и поуздану испоруку нових верзија. Због тога су настали бројни CI/CD алати који аутоматизују процес изградње, тестирања и постављања апликације. Њихов циљ је да открију грешке у раној фази и омогуће континуирани развој.

### 2.1 Jenkins

*Jenkins* је један од најраспрострањенијих CI/CD алата отвореног кода. Његова архитектура се заснива на мастер-агент моделу. У овом моделу мастер управља извршавањем послова и распоређује их на агенте. Агенти могу бити покренути локално или на удаљеним серверима.

Агенти извршавају задатке као што су компилација, тестирање и деплој апликације. Оваква архитектура омогућава расподелу оптерећења и паралелно извршавање послова. Тиме се побољшавају перформансе и скалабилност система.

Комуникација између мастера и агента у *Jenkins*-у се одвија преко *Java Network Launch Protocol* (JNLP) или SSH протокола. JNLP омогућава покретање агента као *Java Web Start* апликације. SSH протокол се користи за удаљене агенте и омогућава сигурнију везу.

Предности *Jenkins*-а су велика флексибилност и подршка за више од хиљаду *plugin*-а. Мане су сложено одржавање, ручна конфигурација и потреба за добрим познавањем система. Безбедност зависи од спољних додатака и често захтева додатну конфигурацију.

### 2.2 GitHub Actions

*GitHub Actions* је део *GitHub* платформе и нуди интегрисано CI/CD решење. У овом систему не постоји класичан мастер-агент однос. Уместо тога, користи се концепт *Runner*-а. Они представљају извршне јединице сличне агентима. *Runner*-и самостално преузимају послове са *GitHub* сервера и извршавају их. Архитектура је једноставнија, али са мањом контролом над процесима.

*Runner*-и могу бити *GitHub Hosted* или *Self-Hosted*. *GitHub Hosted* је *cloud* инстанца коју обезбеђује *GitHub*. Мастер логика је имплицитно интегрисана у *GitHub* платформу. Она управља оркестрацијом и надзором извршавања радних токова.

Радни токови се дефинишу у YAML датотекама унутар репозиторијума. Извршавање радних токова покреће се аутоматски на основу различитих догађаја. Најчешћи догађаји су *push*, *pull request* или креирање новог издања. Сваки радни ток који се извршава на *GitHub Hosted* окружењу покреће се у изолованом виртуелном окружењу. Оваквим приступом се повећава безбедност и стабилност извршавања.

Комуникација се одвија преко *HTTPS REST API*-ја. *Runner*-и периодично шаљу захтеве ка *GitHub* серверу и преузимају послове за извршавање. Овај приступ омогућава сигурну комуникацију, али захтева сталну конекцију.

Предност овог приступа је једноставна конфигурација и чврста интеграција са репозиторијумом кода. Мана је ограничена контрола над инфраструктуром и мања могућност прилагођавања сложеним системима.

## 2.3 GitLab CI/CD

*GitLab* CI/CD интегрише читав *DevOps* процес у једну платформу. Његова архитектура користи мастер-агент модел. *GitLab Server* има улогу мастера, а *GitLab Runner* делује као агент. Мастер управља дефинисаним радним током CI/CD процеса и шаље послове *Runner*-има.

*Runner*-и могу бити локални, удаљени или у *Docker* и *Kubernetes* окружењу. Они преузимају посао од мастера и извршавају задате кораке, као што су *build*, *test* и *deploy*. Након завршетка рада, резултате враћају мастеру. Оваква архитектура омогућава истовремено извршавање више послова. Такође омогућава бољу контролу приступа и једноставније скалирање система.

Комуникација између *GitLab Server*-а и *Runner*-а одвија се преко HTTP(S) протокола. *Runner*-и активно контактирају *GitLab Server* преко API-ја и преузимају послове. Сва комуникација је шифрована путем TLS-а, што обезбеђује сигурност података.

*GitLab* CI/CD је стабилан систем, али за велике пројекте захтева снажну инфраструктуру и пажљиво подешавање.

# Глава 3

---

## Спецификација система

---

У овом поглављу описане су функционалности и технички захтеви система *Test Hub Mini* (назив развијеног система). Систем је осмишљен као дистрибуирани CI/CD алат који омогућава управљање, извршавање и надгледање радних токова у реалном времену.

Архитектура система заснива се на мастер-агент моделу. Мастер је представљен као централизовани систем и координише извршавањем послова. Агенти су извршне јединице задужене за обраду задатака. Систем подржава више типова корака у радном току, као што су клонирање складишта, преузимање и отпремање датотека, извршавање скрипти и генерисање извештаја.

Циљ система је да обезбеди јединствено, сигурно и прошириво решење за континуирану интеграцију и испоруку у дистрибуираним окружењима.

### 3.1 Функционални захтеви

Функционални захтеви дефинишу могућности и понашање система у оквиру дистрибуиране CI/CD архитектуре. Главни циљ је да се омогући стабилан ток рада који подржава:

- управљање корисницима
- агентама
- пословима
- извршавањем
- аналитиком
- интеграцијом са спољним сервисима

#### 3.1.1 Управљање корисницима

Управљање корисницима представља један сегмент система. При иницијалном подизању платформе, систем креира супер администратора. Подаци о овом налогу чувају се у конфигурационом фајлу. Супер администратор може креирати нове администраторе и обичне кориснике. Приликом креирања новог налога систем, генерише иницијалну лозинку која се може променити. Корисницима је омогућено ажурирање личних података, укључујући и промену профилне слике. Безбедност приступа систему обезбеђена је двофакторском аутентикацијом (2FA), преко *Google Authenticator*-а или *Microsoft Authenticator*-а.

#### 3.1.2 Управљање агентима

Управљање агентима представља једну од кључних функционалности система. Систем подржава креирање, измену, брисање и преузимање агента. Сваки агент је задужен за преузимање послова које дефинише мастер и извршавање њихових корака. Систем омогућава лак надзор и управљање статусима агента.

#### 3.1.3 Управљање пословима (*Job*-овима)

Управљање пословима чини једну од основних функционалности система. Сваки посао (*job*) представља радни ток CI/CD процеса и може се састојати од више корака (*step*-ова). Корисници могу креирати нове послове, мењати или брисати постојеће. Приликом покретања посла могу се подесити различити параметри који утичу на извршавање. Систем омогућава праћење извршавања у реалном времену, као и увид у историју свих претходних извршавања. Свака извршена инстанца садржи детаљан приказ статуса појединачних корака и логове који се могу преузети појединачно. У случају да је током извршавања посла дошло до отпремања датотека, систем омогућава преглед и преузимање отпремљених артефаката.

#### 3.1.4 Нотификације и праћење статуса

Систем поседује развијен механизам за обавештавање о статусима послова. За сваку промену статуса, систем шаље нотификацију са информацијом о тренутном стању извршавања. Подржане су три врсте нотификација: *Microsoft Teams*, *Slack* и *In-App* (унутар саме апликације). Корисници могу изабрати за које послове желе да примају обавештења и које типове статуса желе да прате. Подешавање обавештења односи се на *In-App* нотификације, које се могу укључивати или искључивати појединачно за сваки посао и сваки статус. За *Slack* и *Microsoft Teams* нотификације подешавање се врши приликом креирања посла. Корисник сам бира да ли ће посао слати обавештења на ове платформе.

#### 3.1.5 API и интеграције

Ради лакше интеграције са спољним системима, алат подржава генерисање и брисање API кључева. API кључеви омогућавају сигурну комуникацију са другим сервисима. Поред тога, систем подржава *webhook* интеграције за *GitHub* и *GitLab*, што омогућава аутоматско покретање одређених послова на основу активности у репозиторијуму.

#### 3.1.6 Аналитика и извештавање

Систем садржи интегрисани аналитички модул који омогућава праћење активности и понашања корисника у реалном времену. Прикупљају се подаци о географском пореклу корисника, типовима уређаја које користе и најчешћим интеракцијама у систему.



## 3.2 Нефункционални захтеви

### 3.2.1 Безбедност

Систем треба да обезбеди висок ниво заштите података и комуникације између компоненти. Сви пренети подаци морају бити шифровани, а приступ систему ограничен само овлашћеним корисницима. Комуникација између мастера и агента мора бити заснована на сигурним протоколима који гарантују енкрипцију, проверу идентитета и интегритет порука. Поред тога, потребно је осигурати контролу приступа и заштиту спољних интерфејса од неовлашћених захтева.

### 3.2.2 Поузданост

Систем мора бити отпоран на грешке и обезбедити непрекидан рад чак и у случају отказа појединих компоненти. Уколико дође до прекида комуникације или пада агента, остале компоненте морају наставити са радом без утицаја на целокупан процес. Подаци о извршавању послова и статусима морају се чувати на начин који спречава њихов губитак или оштећење.

### 3.2.3 Скалабилност

Архитектура система треба да подржи једноставно проширивање без значајних измена у постојећој структури. Мора бити омогућено додавање нових агената и обрада већег броја послова без смањења перформанси. Систем треба да функционише једнако поуздано у мањим и већим окружењима, уз могућност динамичког прилагођавања оптерећењу.

### 3.2.4 Перформансе

Систем треба да омогући ефикасно извршавање послова и оптимално коришћење ресурса. Обрада података и комуникација између компоненти морају се одвијати без кашњења које би утицало на рад корисника. Распоређивање послова мора бити организовано тако да се избегне преоптерећење појединих чворова и обезбеди равномерна искоришћеност ресурса.

### 3.2.5 Употребљивост

Кориснички интерфејс треба да буде једноставан, прегледан и интуитиван. Све кључне функционалности морају бити лако доступне, а приказ података јасан и разумљив. Систем треба да омогући корисницима лако праћење статуса послова, нотификација и аналитике у реалном времену, као и прилагођавање приказа сопственим потребама.

### 3.2.6 Проширивост

Систем је конципиран тако да се лако може проширити новим функционалностима без већих измена у постојећем коду. Могуће је додати нове типове корака у радним токовима, интеграције са другим сервисима или нове механизме аутентикације. Ова

особина омогућава дугорочно одржавање и прилагођавање специфичним потребама организације.

### 3.2.7 Одрживост и проширивост

Систем мора бити дизајниран тако да омогући лако одржавање и надоградњу. Код и архитектура треба да буду организовани тако да је додавање нових функционалности могуће без значајних измена постојећег решења. Документација мора бити свеобухватна и ажурна, како би се олакшала будућа развојна и интеграциона унапређења.

# Глава 4

---

## Имплементација система

---

У овом поглављу представљена је имплементација дистрибуираног CI/CD система заснованог на мастер-агент архитектури. Поглавље описује начин на који су реализовани функционални и нефункционални захтеви дефинисани у претходном поглављу. Посебан акценат стављен је на структуру система, механизме комуникације, безбедносне компоненте, као и функционалности које омогућавају управљање пословима, агентима и корисницима.

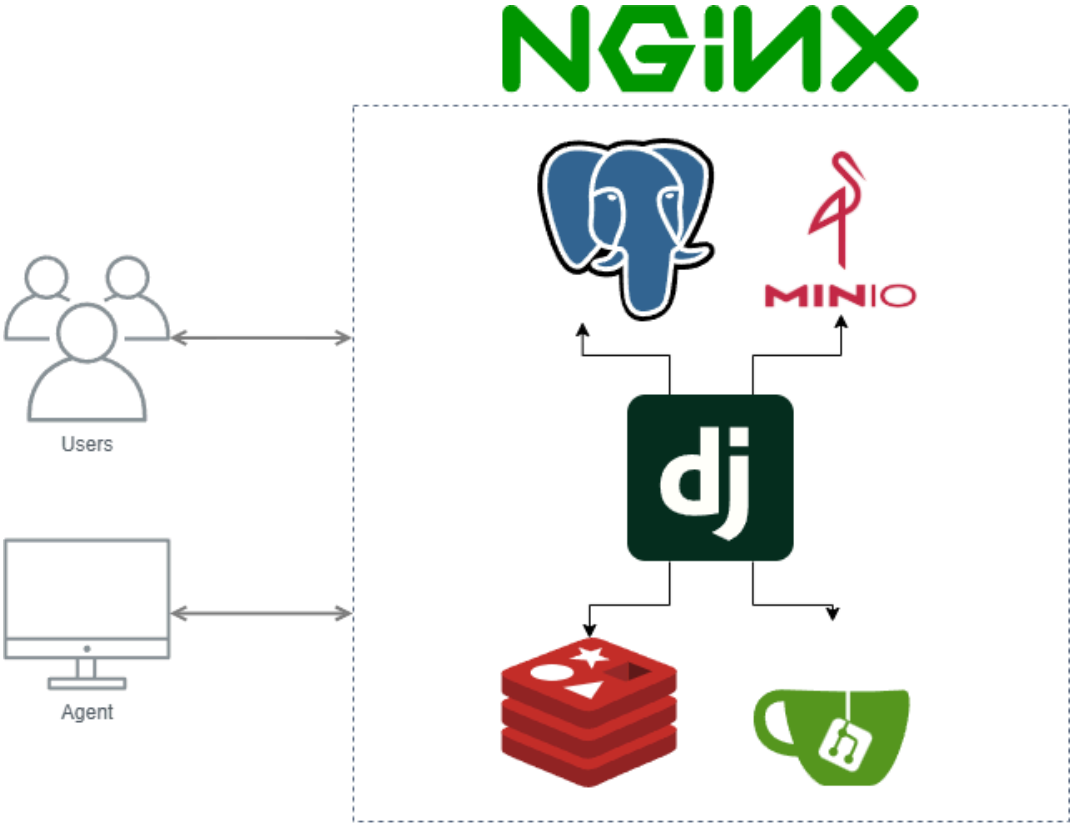
Систем је дизајниран тако да омогући поуздану, безбедну и флексибилну аутоматизацију процеса интеграције и испоруке софтвера. Имплементација обухвата више међусобно повезаних модула, од којих сваки има јасно дефинисану улогу у целокупном процесу. У наредним одељцима биће детаљно описане главне компоненте система, њихове међусобне везе и кључне технологије које су коришћене при развоју.

### 4.1 Архитектура система

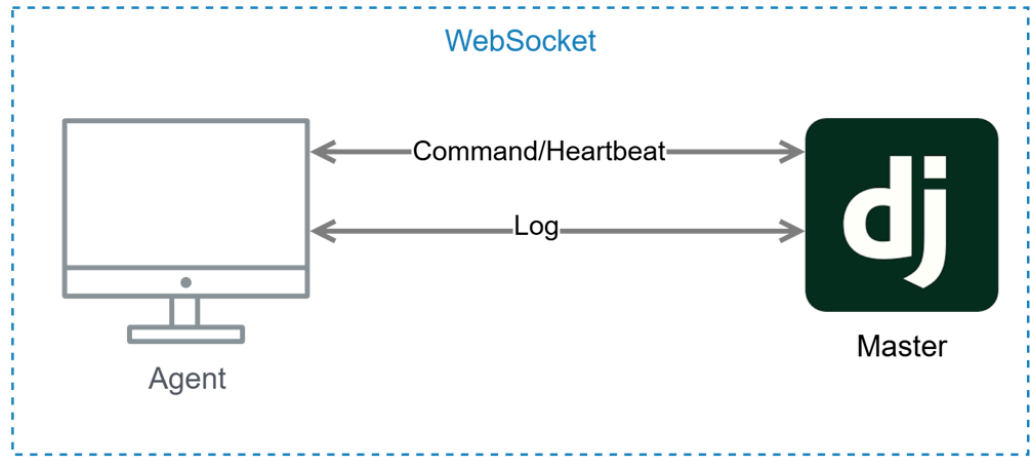
#### 4.1.1 Преглед архитектуре

На Слици 1 приказана је архитектура система која се заснива на мастер-агент моделу. Мастер представља централну компоненту која управља свим процесима у систему. Он се повезује са пратећим сервисима, као што су база података (*PostgreSQL*), систем за кеширање (*Redis*), складиште артефаката (*MinIO*) и интерфејс за корисничку комуникацију. *Nginx* делује као посредни слој који обезбеђује сигуран приступ и усмеравање захтева ка мастер серверу. Агенти се повезују са системом преко сигурне *WebSocket* везе и извршавају задатке које мастер дефинише.

На слици 2 приказан је механизам комуникације између мастер-а и агената. Комуникација се заснива на *WebSocket* протоколу који омогућава двосмерну размену података у реалном времену. За потребе рада система користе се два одвојена *WebSocket* канала. Први канал служи за слање команди и периодичних сигнала (*heartbeat*). Други канал користи за пренос логова и извештаја о извршавању послова. Комуникација је подељена у два канала како би се избегло загушење и осигурао стабилан проток података. Оваква организација обезбеђује поуздану синхронизацију и непрекидан рад система.



Слика 1: Архитектура система



Слика 2: Комуникација између мастер-а и агента

#### 4.1.2 Мастер компонента

Мастер представља централни део система и имплементирана као *Django* апликација. Задужен је за управљање процесима и оркестрацију послова. Он координише рад свих агената и контролише извршавање задатака које дефинишу корисници.

Кориснички интерфејс развијен је као апликација која комуницира са мастером преко *RESTful API*-ја. Сви захтеви и одговори размењују се у *JSON* формату. Интерфејс омогућава корисницима да управљају свим ресурсима система и прате извршавање послова у реалном времену.

Приликом креирања посла, корисник бира агента који ће га извршити. Мастер прослеђује дефинисани посао изабраном агенту. Агент преузима тај посао и извршава га. Уколико је изабрани агент тренутно заузет, мастер ставља посао у ред чекања. Послови који се већ налазе у реду имају предност у односу на нове захтеве, чиме се обезбеђује фер редослед извршавања.

Мастер одржава централизовану евиденцију свих дефинисаних послова и њихових статуса, као и информације о агентима и њиховој доступности. Не учествује у самом извршавању, већ управља процесом и надгледа комуникацију између корисника и агената.

Поред управљања пословима, мастер обрађује све захтеве који стижу из корисничког интерфејса. Корисници преко интерфејса могу да креирају, мењају или бришу послове, као и да управљају агентима. Мастер верификује аутентичност захтева, примењује безбедносне политике и прослеђује инструкције одговарајућим сервисима.

#### 4.1.3 Агент компонента

Агент представља извршну компоненту система задужену за обраду послова које шаље мастер. Реализован је као мала *Python* апликација која се изграђује и пакује у *.exe* формат. На тај начин агент се може лако преузети и покренути на било ком рачунару, без потребе за додатним подешавањима или инсталацијом зависности.

Сваки агент ради независно и може бити инсталиран на различитим машинама или унутар изолованих окружења. Оваква организација омогућава равномерну расподелу оптерећења и паралелно извршавање више послова.

Агент периодично шаље *heartbeat* сигнале мастеру како би потврдио своју доступност. На основу тих сигнала, мастер прати активност свих агената у систему. Уколико се *heartbeat* не прими у предвиђеном времену, агент се означава као неактиван. Његови послови се аутоматски паузирају и настављају када се веза поново успостави.

Када агент прими посао, преузима податке које је дефинисао мастер и покреће извршавање корака. Током процеса агент шаље логове и статус сваког корака, што омогућава корисницима да у реалном времену прате напредак извршавања.

Оваква архитектура рада агената обезбеђује поуздано извршавање послова, лако скалирање и једноставно одржавање целокупног система.

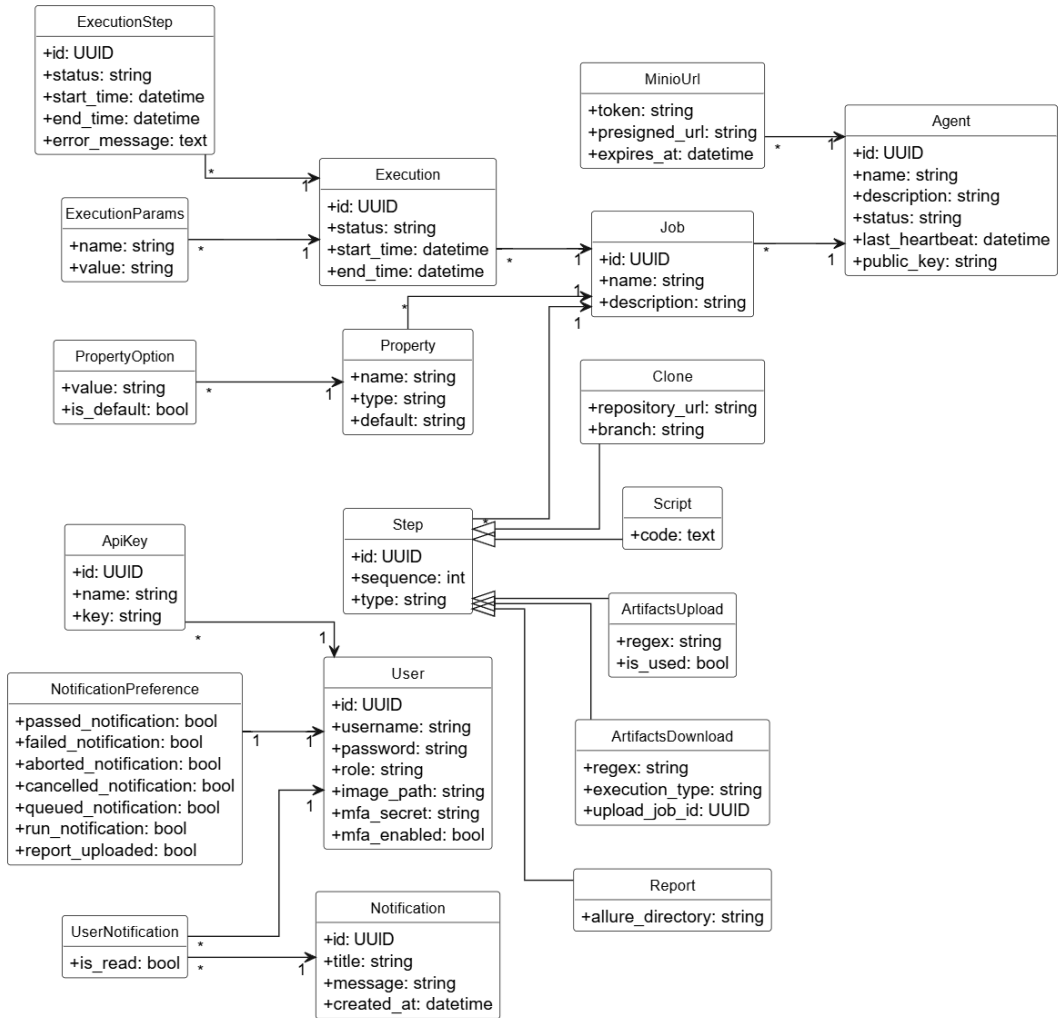
### 4.1.4 База података

База података представља централно место за чување свих података у систему. Користи се *PostgreSQL* као релациона база података. Изабрана је због стабилности, поузданости и добре подршке за рад у вишекорисничким и дистрибуираним окружењима. Сви подаци који се односе на кориснике, агенте, послове, извршења и нотификације чувају се у овој бази, док су артефакти и већи бинарни фајлови смештени у засебном складишту (*MinIO*).

Главни ентитети у систему су:

- *User* – представља корисника система и садржи податке о налогу, улози, слици и стању двофакторске аутентикације.
- *Agent* – представља извршну јединицу задужену за обраду послова. Чува податке о статусу, кључевима и времену последњег *heartbeat*-а.
- *Job* – дефинише посао који се извршава преко изабраног агента. Садржи основне информације и повезане кораке.
- *Step* – описује појединачни корак у извршавању посла.
- *Execution* и *ExecutionStep* – чувају податке о историји извршавања послова и статусима сваког корака.
- *Notification* и *UserNotification* – користе се за слање и евидентирање нотификација унутар система.
- *Property* и *PropertyOption* – омогућавају конфигурацију при покретању послова, са дефинисаним типовима и подразумеваним вредностима.
- *ApiKey*, *GitHubInstallation* и *GitHubAppInstallState* – служе за интеграцију са спољним системима и безбедан приступ преко API интерфејса.

На Слици 3 приказан је поједностављени дијаграм класа који илуструје структуру и односе између главних ентитета система.

Слика 3: Дијаграм класа система *Test Hub Mini*

---

## **4.2 Безбедносни механизми**

### **4.2.1 Двофакторска аутентикација (2FA)**

### **4.2.2 Контрола приступа и корисничке улоге**

### **4.2.3 Шифровање и сигурна комуникација**

### **4.2.4 Nginx као reverse proxy и сигурносни слој**

## **4.3 Управљање корисницима и агентима**

### **4.3.1 Креирање и администрирање налога**

### **4.3.2 Управљање агентима**

## **4.4 Управљање пословима (Job-овима)**

### **4.4.1 Креирање и конфигурисање послова**

### **4.4.2 Извршавање послова**

### **4.4.3 Историја извршавања и артефакти**

## **4.5 Систем нотификација**

### **4.5.1 In-App нотификације**

### **4.5.2 Slack и Microsoft Teams интеграције**

## **4.6 Интеграције са спољним системима**

### **4.6.1 Подршка за GitHub и GitLab webhook-ове**

### **4.6.2 Генерисање и управљање API кључевима**

## **4.7 Корисничка аналитика**

### **4.7.1 Сакупљање и обрада података о корисницима**

### **4.7.2 Интеграција са Umami и PostHog системима**



## Глава 5

---

### Закључак

---

У закључку дајте кратак преглед онога шта урађено, са освртом на проблеме који су решени, предности и мане решења и правце даљег развоја.

---

---

## Списак слика

---

Слика 1	Архитектура система .....	10
Слика 2	Комуникација између мастер-а и агента .....	10
Слика 3	Дијаграм класа система <i>Test Hub Mini</i> .....	13

---

---

## **Списак листинга**

---

---

---

## **Списак табела**

---

---



---

## Биографија

---

Никола Трајковић је рођен 19. децембра 2002. године у Врању. Основну школу „Доситеј Обрадовић“ у Врању завршио је 2017. године као носилац Вукове дипломе. Гимназију „Бора Станковић“ у Врању завршио је 2021. године као носилац Вукове дипломе. Исте године уписује Факултет техничких наука у Новом Саду, одсек Софтверско инжењерство и информационе технологије. Положио је све испите предвиђене планом и програмом.

---

---

## **Литература**

---