

**ADVANCED DEVELOPMENT OF BACKEND APPLICATION FOR SMART
GRID CLUSTER CONTROLLER**

By

Gregorius Ferdinand
11902004

BACHELOR'S DEGREE
in

INFORMATION TECHNOLOGY
FACULTY OF ENGINEERING AND INFORMATION TECHNOLOGY

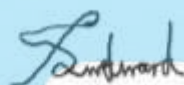


SWISS GERMAN UNIVERSITY
The Prominence Tower
Jalan Jalur Sutera Barat No. 15, Alam Sutera
Tangerang, Banten 15143 - Indonesia

Revision after the Thesis Defense on 10 July 2023

STATEMENT BY THE AUTHOR

I hereby declare that this submission is my own work and to the best of my knowledge, it contains no material previously published or written by another person, nor material which to a substantial extent has been accepted for the award of any other degree or diploma at any educational institution, except where due acknowledgement is made in the thesis.



Gregorius Ferdinand

Student

24.07.2023

Date

Approved by:



Prof. Dr.-Ing. Egon Ortjohann

Thesis Advisor

21.07.2023

Date



Dipl.-Inf. I Eng Kho

Thesis Co-Advisor

24.07.2023

Date

Dr. Maulahikmah Galinium, S.Kom., M.Sc.

Dean

Date

Gregorius Ferdinand

ABSTRACT

ADVANCED DEVELOPMENT OF BACKEND APPLICATION FOR SMART GRID CLUSTER CONTROLLER

By

Gregorius Ferdinand
Prof. Dr.-Ing. Egon Ortjohann, Advisor *
Dipl.-Inf. I Eng Kho, Co-Advisor **

SOUTH WESTPHALIA UNIVERSITY OF APPLIED SCIENCES *
SWISS GERMAN UNIVERSITY **

Smart grid correlates to the maintainability of the reliability and sustainability of electric grids. Smart grid was introduced alongside the Clustering Power System Approach (CPSA) with an aim to balance power systems deployed in local areas. Utilizing Information Technology (IT) in smart grid development, the Smart Grid Cluster Controller (SGCC), a realization of CPSA, utilizes the Linux Operating System (OS). A software application regarding data manipulation within the SGCC is required to monitor and manage the grids. This thesis covers the further development of the existing backend application (server-side) that is necessary for data manipulation as well as handling requests from users, acting as a bridge between the client side and the database. The requests will take the form of a Representational State Transfer (REST) Application Programming Interface (API) endpoint where users are required to input their desired request variables. This is developed with the Node.js server environment which is based on JavaScript, as well as the Express framework.

Keywords: Smart Grid Cluster Controller, Backend, Database, Data Manipulation, Node.js, Real-Time Data.



© Copyright 2023
by Gregorius Ferdinand

All rights owned by the South Westphalia University of Applied Sciences,
Institute for Electrical Power Systems,
Under the law of research projects from the
“Ministry of Education and Research”,
Federal Republic of Germany

DEDICATION

I dedicate this work for my family, friends, mentors, and co-workers, for all their endless guidance and support throughout my entire life.



ACKNOWLEDGEMENTS

First and foremost, I would like to thank God almighty for the path that he has given me throughout my life. Without his blessing and holy grace, I would have not made it this far and I would not be the person I am today.

Secondly, I would like to thank Prof. Dr.-Ing. Egon Ortjohan for the opportunity to be involved in this thesis at his laboratory and for his guidance throughout the development of this thesis. I would like to thank Dipl.-Ing. Andreas Schmelter and my senior Joseph Andreas, S.Kom., B.Eng., M.Sc. for all their encouragement and guidance throughout the thesis work. Without their support, this thesis would have not been possibly completed. I would like to thank my co-advisor Dipl.-Inf. I Eng Kho for his guidance during the thesis writing, and the lecturers from the Information Technology department, Dr. Maulahikmah Galinium, S.Kom., M.Sc., Mr. James Purnama, S.Kom., M.Sc., and Dipl.-Ing. Randy Anthony, S.Kom., M.Kom.. For their patience and all of the knowledge and opportunities that they have given me throughout my studies. I would like to express my deepest gratitude to my senior Leonardo Theoson, S.Kom., B.Eng. and everyone at CodeDoc Indonesia for all of their guidance and support throughout my work, as well as being a great inspiration to myself.

I would like to thank Arkent Nathanael, Muhammad Raka Zuhdi Harypradana, Dave Andersen, and my friends in the Information Technology program for all the memories we have made together, it has been a unique and memorable experience, be it academic or non-academic. I wouldn't be where I am without their support. As well as the entire batch of SGU 2019 students for four years of experience.

Lastly, I would like to thank my parents, my sister, my dog, and my entire family for always being there for me and their endless support.

TABLE OF CONTENTS

	Page
STATEMENT BY THE AUTHOR.....	2
ABSTRACT.....	3
DEDICATION.....	5
ACKNOWLEDGEMENTS.....	6
TABLE OF CONTENTS.....	7
LIST OF FIGURES.....	10
LIST OF TABLES.....	12
LIST OF SCRIPTS.....	13
CHAPTER 1 - INTRODUCTION.....	14
1.1. Background.....	14
1.2. Research Problems.....	16
1.3. Research Objectives.....	16
1.4. Research Scope.....	17
1.5. Research Limitation.....	17
1.6. Significance of Study.....	18
1.7. Research Questions.....	18
1.8. Hypothesis.....	19
CHAPTER 2 - LITERATURE REVIEW.....	20
2.1. Smart Grid.....	20
2.2. Information Technology in the Development of Smart Grid.....	21
2.3. Clustering Power System Approach.....	21
2.3.1. Smart Grid Cluster Controller.....	23
2.4. Authorization and Authentication for User Rights.....	23
2.5. Backend Development Environment.....	24
2.5.1. Linux as SGCC Operating System.....	25
2.5.2. PuTTY for Linux Terminal Virtualization.....	26
2.5.3. Docker to Containerize Applications.....	27
2.6. Programming Development Environment.....	28
2.6.1. JavaScript for Server-side Development.....	28
2.6.2. Node.js as Server Environment.....	29
2.6.3. Express Framework for Node.js Backend Development.....	31
2.6.4. JavaScript Object Notation as Data Format.....	31
2.6.5. JSON Web Token.....	32

2.6.6.	Promise and Semaphores.....	34
2.7.	Database Management System	35
2.7.1.	ArangoDB as Multi-Model Database.....	35
2.7.2.	InfluxDB as Time-Series Database	36
2.8.	Web Access Environment.....	38
2.8.1.	Hyper Text Transfer Protocol as Communication Protocol.....	38
2.8.2.	Representational State Transfer Application Programming Interface Endpoint	39
2.9.	Software Testing Environment	40
2.9.1.	Postman as Manual Software Testing Environment	40
2.9.2.	Shell Scripting for Automated Software Testing	41
2.9.3.	Apache JMeter for Simulated Software Testing	42
2.10.	Waterfall Methodology for Software Development	43
2.11.	Related Works Regarding this Thesis	45
2.11.1.	Backend System Development for Kunyahku Online Catering Startup as a Case Study using Node.js	46
2.11.2.	Development of Node.js based Backend System with Multiple Storefronts for Batik Online Store.....	46
2.11.3.	An API first Methodology for Designing a Microservice-based Backend as a Service Platform	47
2.11.4.	Further Development and Implementation of Browser-Based User Interface for Smart Grid Cluster Controller	48
2.11.5.	Development of Web-Based User Interface for Smart Grid Real- Time Orchestration System	48
2.12.	Research Comparison	49
CHAPTER 3 - RESEARCH METHODS		51
3.1.	Implementation of Waterfall Methodology	51
3.2.	Planning and Analysis.....	51
3.3.	Software Design.....	61
3.4.	Software Development.....	64
3.5.	Software Testing	64
3.5.1.	Function Testing.....	65
3.5.2.	Stress Testing	65
CHAPTER 4 - RESULT AND DISCUSSION		66
4.1.	Real-Time Data Exchange Issue	66
4.2.	Communication Overhead in Data Objects Exchange.....	68
4.3.	Utilizing Promise Object for Real-Time Data Issue	70
4.4.	Implementation of Batch-create Service with User Authentication and Authorization	73
4.5.	Software Testing Results	76
4.5.1.	Function Testing.....	76
4.5.2.	Stress Testing	82

4.6. Result Analysis	88
4.6.1. Concurrent Real-Time Data Exchange.....	88
4.6.2. Multiple Data Objects Exchange with User Authentication and Authorization.....	90
CHAPTER 5 - CONCLUSIONS AND RECOMMENDATIONS	94
5.1. Conclusions.....	94
5.2. Future Recommendations	95
GLOSSARY	96
REFERENCES	98
CURRICULUM VITAE.....	103



LIST OF FIGURES

Figures	Page
Figure 1.1: Smart Grid Market Size Prediction Chart (Precedence Research, 2021)..	15
Figure 2.1: Smart Grid Architecture (Trilliant Holdings Inc., 2012).....	20
Figure 2.2: Clustering Power System Approach Overview (Ortjohann <i>et al.</i> , 2011)..	22
Figure 2.3: Smart Grid Cluster Controller Logo (Power Supply Department at Fachhochschule Südwestfalen, 2020)	23
Figure 2.4: Tux, Logo and Mascot of Linux (Loshin and Bigelow, 2022).....	26
Figure 2.5: Example of PuTTY Terminal (SSH Communications Security Inc, 2023)	27
Figure 2.6: Docker Logo (Press and Media Resources - Docker, 2022)	28
Figure 2.7: JavaScript Logo (Williams, 2022).....	29
Figure 2.8: Node.js Logo (Node.js, 2022)	30
Figure 2.9: NPM Logo (NPM, 2023).....	30
Figure 2.10: Express Logo (expressjs, 2023).....	31
Figure 2.11: Example of a JSON Document (ConvertSimple, 2023).....	32
Figure 2.12: ArangoDB Logo (ArangoDB, 2023).....	36
Figure 2.13: InfluxData Logo (InfluxData, 2023)	36
Figure 2.14: Postman Logo (Postman Store, 2023)	41
Figure 2.15: Apache JMeter Logo (Apache Software Foundation, 2022).....	43
Figure 2.16: Waterfall Methodology (Ionos, 2019).....	44
Figure 3.1: SGCC Application Architecture (Power Supply Department at Fachhochschule Südwestfalen, 2020)	52
Figure 3.2: Backend Architecture Diagram (Power Supply Department at Fachhochschule Südwestfalen, 2020)	53
Figure 3.3: Use Case Diagram of the SGCC Backend System (Power Supply Department at Fachhochschule Südwestfalen, 2020)	54
Figure 3.4: Login Activity Diagram (Power Supply Department at Fachhochschule Südwestfalen, 2020).....	55

Figure 3.5: Create Data Activity Diagram (Power Supply Department at Fachhochschule Südwestfalen, 2020)	56
Figure 3.6: View (Read) Data Activity Diagram (Power Supply Department at Fachhochschule Südwestfalen, 2020)	57
Figure 3.7: Update Data Activity Diagram (Power Supply Department at Fachhochschule Südwestfalen, 2020)	58
Figure 3.8: Delete Data Activity Diagram (Power Supply Department at Fachhochschule Südwestfalen, 2020)	60
Figure 3.9: Concurrent Request Activity Diagram	62
Figure 3.10: Batch-create Data Activity Diagram	63
Figure 4.1: One of the Response Results with "null"	67
Figure 4.2: Average Duration of Single Create Service (ms)	68
Figure 4.3: Flow Chart for Concurrent Requests Service	71
Figure 4.4: Flowchart of Read Data Function From InfluxDB	72
Figure 4.5: Flowchart for Batch-create Service to Create and Send Multiple Data Objects	73
Figure 4.6: Flowchart of Create Data Function to ArangoDB (Power Supply Department at Fachhochschule Südwestfalen, 2020)	74
Figure 4.7: Flowchart of Verifying Request Size	75
Figure 4.8: Different Concurrent Request Results (“ping measurement”)	78
Figure 4.9: Different Concurrent Request Results (“kernel measurement”)	79
Figure 4.10: Responses of the While(true) Shell Script.....	84
Figure 4.11: Response as shown in a Log File Within the 8000 Kilobytes Range.....	85
Figure 4.12: Request Timed Out Error as Shown in a Log File with 1 Kilobyte	85
Figure 4.13: Average Duration of Batch-create Service.....	86
Figure 4.14: Comparison of Average Response Size (in bytes) of the Concurrent Request Service.....	88
Figure 4.15: Comparison of Average Duration (ms) between Single-create and Batch-create Service Implementation.....	90
Figure 4.16: Successful Request Message	92
Figure 4.17: Permission Denied Message.....	92
Figure 4.18: Request Size Too Large Error	93

LIST OF TABLES

Table	Page
Table 2.1: Development Environment Used for Backend Development.....	24
Table 2.2: JSON Web Token Structure (jwt.io, 2023).....	33
Table 2.3: Promise States (Mozilla, 2023).....	34
Table 2.4: Database Structure of InfluxDB (InfluxData, 2023)	37
Table 2.5: HTTP Status Codes (Cloudflare, 2022).....	38
Table 2.6: HTTP Methods and Respective CRUD Functions (Codecademy, 2023)...	40
Table 2.7: Publications of Related Works	45
Table 2.8: Comparison Table.....	49
Table 4.1: Average Duration of Single Create Service.....	69
Table 4.2: Test Cases for Concurrent Requests Service	76
Table 4.3: Test Cases for Batch-create Service	81
Table 4.4: Average Duration of Batch-create Service	87
Table 4.5: Comparison of Average Response Size (in bytes) of the Concurrent Requests Service	89
Table 4.6: Comparison of Average Duration (ms) Before and After Batch-create Service Implementation	91

LIST OF SCRIPTS

Script	Page
Script 2.1: Example of a Shell Script.....	42
Script 4.1: Shell Script Used to Test the Concurrent Request Service	66
Script 4.2: Shell Script for Different Concurrent Requests	77
Script 4.3: Shell Script Used to Test the Batch-create Service.....	80
Script 4.4: Shell Script Used for Concurrent Requests Stress Testing	82

