PAPEL • **ACESSO ABERTO**

Projeto e Implementação da API REST para Sistema de Informação Acadêmica

Para citar este artigo: AA Prayogi et al 2020 IOP Conf. Ser.: Mater. ciência Eng. **875** 012047

Projeto e Implementação da API REST para Acadêmico Sistema de informação

AA Prayogi1*, M Niswar1 , Indrabayu1 e M Rijal1

1Departamento de Engenharia Informática, Faculdade de Engenharia, Universidade Hasanuddin, Makassar, Indonésia

*E-mail: aisprayogi@unhas.ac.id

Resumo. Com o aumento do número de sistemas de informação utilizados em uma organização, aumenta também a importância da troca de dados entre esses sistemas. Esta pesquisa trata do desenvolvimento de protótipo e análise de desempenho da API Rest para sistema de informação acadêmica.
A Rest API foi desenvolvida usando duas tecnologias de servidor diferentes, NodeJS e PHP. O protótipo foi desenvolvido em cima de um servidor de banco de dados utilizando uma tabela exemplo que representa um funcionário de uma instituição de ensino superior. Para cada API Rest desenvolvida, foram criados 2 tipos de endpoint. O experimento foi definido com um banco de dados contendo uma única tabela e utilizou o Apache Jmeter para simular até 1.000 solicitações simultâneas. Os resultados do experimento mostram que a implementação NodeJS da API REST tem consistentemente melhor desempenho em comparação com a implementação da API REST baseada em PHP. A implementação do NodeJS atingiu 100% de throughput para até 1.000 solicitações simultâneas, enquanto o PHP atingiu 48,70% de throughput ao atender o mesmo número de solicitações simultâneas.

1. Introdução O

uso de sistema de informação, especialmente sistema de informação baseado na web, no campo acadêmico, especialmente em uma instituição de ensino superior, tornou-se a norma. O sistema de informação desempenha um papel importante nos vários processos de negócios em uma instituição de ensino superior, que vão desde o processo de matrícula de novos alunos, matrícula de curso no início de um período letivo, processo de avaliação de final de período letivo até o processo de gerenciamento de informações de graduação e ex-alunos [1].

Para acomodar e garantir que o fluxo desses processos de negócios funcione sem problemas, também existem vários e diversos sistemas de informação configurados. Alguns sistemas de informação mais comuns usados por uma instituição de ensino superior, incluindo Learning Management System (LMS), Content Management System (CMS), Asset Management System, Student Registration System e Course Management System. Na maioria das instituições, esses diferentes tipos de sistemas de informação são desenvolvidos internamente (por unidade de trabalho de tecnologia da informação), terceirizados para um fornecedor terceirizado ou personalizados a partir de projetos comunitários de código aberto. A manutenção de um aplicativo baseado em solução proprietária parte do próprio fornecedor. O processo para desenvolver um novo recurso ou para corrigir certos bugs requer um longo processo administrativo que fica para trás do cronograma do usuário. A solução personalizada baseada em código aberto também enfrenta o mesmo problema, em que cada bug ou novo recurso deve ser registrado primeiro antes de poder ser process
Essas circunstâncias levam a novos recursos ou correção de bugs que ficam aquém das expectativas e necessidades do usuário [2].

Dentro da instituição existe a necessidade de haver uma comunicação de dados comum e uniforme entre cada sistema de informação. O objetivo dessa comunicação uniforme de dados é evitar a duplicação e incompatibilidade de dados armazenados em cada sistema de informação. Com os dados uniformes

formato de comunicação, novas aplicações podem ser construídas de forma mais fácil e rápida usando dados armazenados nos sistemas previamente estabelecidos. O número de uso de smartphones entre os usuários da Internet tem aumentado constantemente a cada ano. Os usuários de smartphones na Indonésia atingiram 83,5 milhões de usuários em 2018. Embora a maioria dos sistemas de informação seja facilmente acessada a partir do navegador móvel, o uso de aplicativos nativos móveis tem certas vantagens sobre o site móvel [3]. Os aplicativos nativos móveis precisam se comunicar com o servidor do sistema de informações existente por meio da Interface de Programação de Aplicativos (API).

Existem muitos tipos de arquitetura que podem ser escolhidos para construir a comunicação de dados entre sistemas de informação ou aplicativos. Uma das opções é usar a arquitetura da API REST (Representational State Transfer Protocol). Com a arquitetura REST, o serviço da Web pode ser criado usando métodos simples e a arquitetura HTTP subjacente. Um protótipo de arquitetura API REST para sistema de informação acadêmica construído com dois tipos de tecnologias de framework. Os protótipos então analisaram o desempenho com três parâmetros (tempo de resposta, taxa de transferência e perda de pacotes) em várias conexões simultâneas.

2. API REST

REST (Representational state transfer) é uma API arquitetônica (Application Programming Interface) que fornece comunicações cliente-servidor para aplicações Web sobre o protocolo HTTP, tornando-o facilmente implementado, pois não está vinculado a nenhum protocolo de transferência. Os três principais princípios de design do REST são endereçamento, interface uniforme e ausência de estado [4]. O REST aborda a aceitabilidade definindo pontos de extremidade em uma estrutura de diretório [5] por meio de URI diferente para extrair os dados. A API funciona com base no princípio CRUD (Create, Read, Update, Delete), que corresponde às funções mais populares [4] INSERT, SELECT, UPDATE e DELETE, em armazenamentos de dados persistentes, como SQL.

Chamar um serviço RESTful pelo protocolo HTTP pode ser feito em várias linguagens de programação.
Em jQuery, um framework de Javascript, um servidor REST pode ser chamado a partir de uma consulta Ajax. Uma resposta comum de dados está na forma de dados JSON (Javascript Object Notation). A resposta JSON típica pode ser vista na Figura 1. Os dados também podem ser retornados na forma de XML (eXtensible Markup Language) [4].

Figura 1. Exemplos de resposta JSON

3. Tecnologias do lado do servidor

O desenvolvimento de serviços da Web com arquitetura REST API atualmente pode ser feito com várias tecnologias de servidor. Decidiu-se duas tecnologias de servidor mais utilizadas, que são NodeJS e PHP.

3.1. NodeJS

Node JS é uma tecnologia de servidor web baseada em JavaScript. Ele usou o V8 Javascript Engine com complementos de módulo integrados. O NodeJS usou arquitetura orientada a eventos para lidar com solicitações de clientes. Com a arquitetura orientada a eventos, todas as solicitações dos clientes serão colocadas na fila de eventos. O loop de eventos de encadeamento único do NodeJS, em seguida, escolha uma das solicitações para verificar se a solicitação requer bloqueio de operação de E/S ou tarefas de computação complexas. Se for, a solicitação será movida para um thread diferente obtido do NodeJS Internal Pool Thread [6]. Depois que o thread terminar de processar a solicitação, bloqueando a operação de E/S ou tarefas de computação complexas, ele enviará a resposta de volta à fila de eventos para ser processada pelo loop de eventos de thread único do NodeJS posteriormente. Este processo é mostrado na Figura 2.

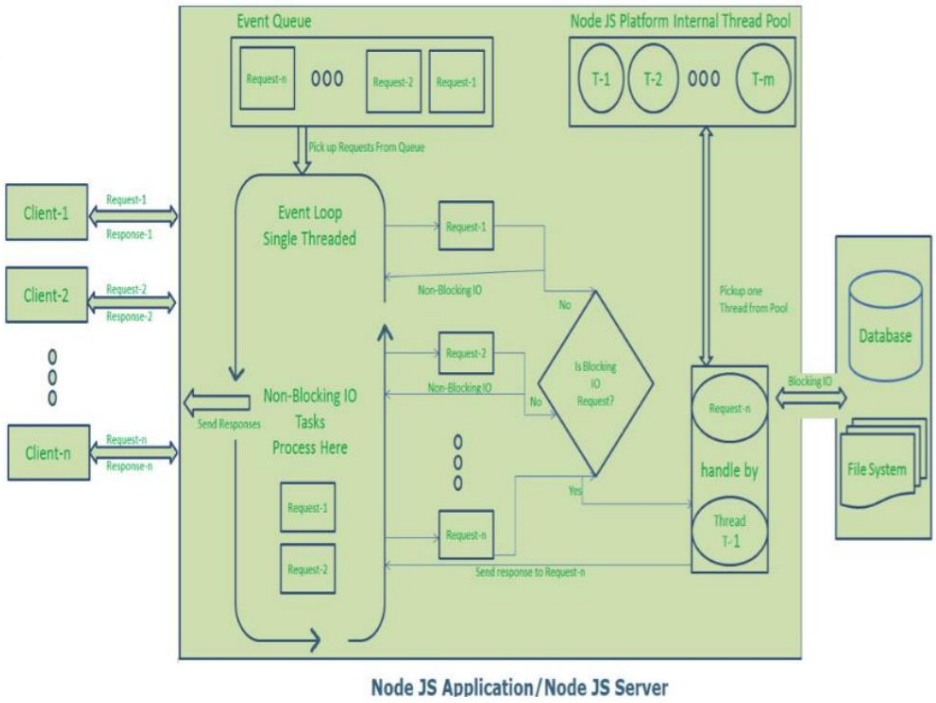


Figura 2. Arquitetura NodeJS [6]

3.2. PHP

PHP também conhecido como PHP:Hypertext Preprocessor é uma linguagem de script desenvolvida para adicionar dinamismo à página HTML estática. Foi desenvolvido pela primeira vez em 1994 por Rasmus Lerdorf. O script PHP é processado por um determinado processo chamado interpretador PHP em um servidor da Web para gerar código HTML ou dados de imagem binária. PHP evoluiu de programação de paradigma procedural para programação de tipo orientada a objetos em sua última iteração (PHP 7).

4. Experimento O
experimento foi projetado para mostrar como as duas tecnologias de servidor usadas para implementar a API REST funcionaram em várias conexões simultâneas. Existem 2 tipos de medição de desempenho usados neste experimento, incluindo tempo de solicitação e taxa de transferência. O tempo de solicitação mede o tempo decorrido entre o momento em que o cliente envia a solicitação até o cliente receber respostas do servidor. Taxa de transferência é o número total de solicitações processadas em determinado período de tempo fixo. A perda de pacotes é medida como a porcentagem de pacotes perdidos em relação aos pacotes enviados.

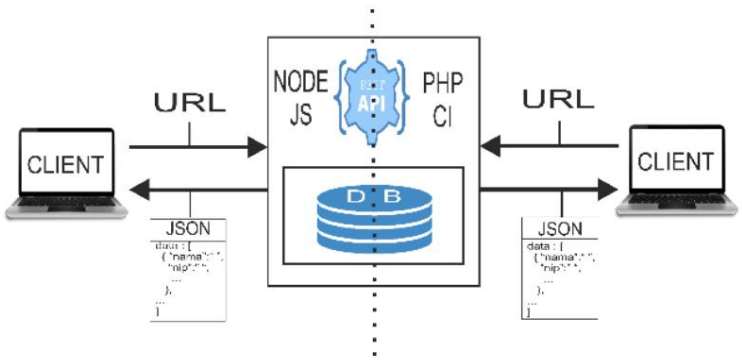
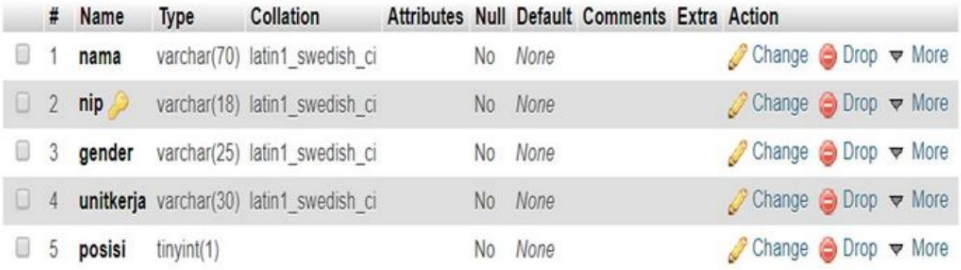


Figura 3. Arquitetura do protótipo

O protótipo foi construído conforme mostrado na figura 3. Os clientes enviam solicitações acessando determinadas URLs fornecidas pelo servidor. O servidor foi configurado com duas tecnologias de servidor, NodeJS e PHP com Apache como servidor web. Toda vez que a solicitação é recebida por NodeJS ou PHP, a solicitação é processada com base no parâmetro e nos dados incorporados pela tecnologia de servidor da Web respeitada e retorna a resposta no formato JSON (Javascript Object Notation).

Um banco de dados foi configurado no servidor para servir como repositório de dados para a implementação da API REST NodeJS e PHP. O banco de dados contém uma tabela, denominada tabela *profilpegawai* . A tabela contém 5 campos e 3.920 linhas de dados, copiados do atual sistema de gerenciamento de informações de funcionários implantado na Hasanuddin University. A estrutura detalhada da tabela pode ser vista na Figura 4.



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	nama	varchar(70)	latin1_swedish_ci		No	None			Change Drop More
2	nip	varchar(18)	latin1_swedish_ci		No	None			Change Drop More
3	gender	varchar(25)	latin1_swedish_ci		No	None			Change Drop More
4	unitkerja	varchar(30)	latin1_swedish_ci		No	None			Change Drop More
5	posisi	tinyint(1)			No	None			Change Drop More

Figura 4. Detalhes da estrutura da tabela *profilpegawai*

Para este experimento, dois tipos de endpoints da API REST foram criados, conforme mostrado na tabela 1. O primeiro endpoint é usado quando um cliente deseja solicitar todas as linhas, neste caso, dados do funcionário, da tabela *profilpegawai* . O segundo terminal é usado para um cliente solicitar determinados dados de funcionários que correspondam ao parâmetro NIP. Ambas as respostas de endpoint são retornadas no formato JSON.

Tabela 1. Terminais da API REST

Parâmetro	URLs	
	NodeJS	PHP
OBTER TUDO	http://localhost:3000/profilpegawai/	http://localhost:8080/rest-api/rest ci/api/profilpegawai http://
PARAMS	http://localhost:3000/profilpegawai/:nip	localhost:8080/rest-api/rest GET ci/api/profilpegawai ?nip=NIP

Depois de configurar o banco de dados e os endpoints da API, a próxima etapa foi preparar a simulação para o cenário de teste. Apache JMeter foi usado para simular solicitações de clientes. Para medir o tempo de resposta, para cada implementação de NodeJS e PHP, foram feitas 10 solicitações para cada endpoint. Cada requisição foi repetida em 30 iterações e então foi calculada a média do tempo de resposta. A taxa de transferência foi medida simulando solicitações de 1, 100, 200, 300, 400, 500, 600, 700, 800, 900 e 1000 solicitações para cada endpoint em cada implementação NodeJS e PHP.

5. Resultado e Discussão O

tempo de resposta para a implementação da API REST NodeJS mostra um resultado significativamente melhor do que a implementação do PHP. Ao testar o endpoint GET ALL, o tempo médio de resposta para NodeJS variou de 138ms a 156ms, enquanto a implementação do PHP variou de 244ms a 266ms. Os resultados completos da medição do tempo de resposta para o endpoint GET ALL são mostrados na tabela 2. Resultado semelhante também foi mostrado para o endpoint GET PARAMETER, no qual o NodeJS mostra melhores resultados do que sua contraparte PHP.

Tabela 2. Resultados do tempo de resposta para terminal GET ALL

Solicitar	PHP			NODEJS		
	máximo (ms)	min (ms)	média (ms)	máximo (ms)	min (ms)	média (ms)
1	894	207	266	188	113	138
2	446	206	253	371	107	146
3	369	208	244	503	113	156
4	795	211	257	212	116	141
5	410	207	248	159	110	136

Durante a medição da taxa de transferência, a implementação do NodeJS tem um desempenho melhor em comparação com a implementação do PHP. Ao usar o endpoint GET ALL, a implementação do NodeJS é consistentemente capaz de lidar com todas as solicitações de 1 a 1.000 solicitações simultaneamente. A implementação do PHP, por outro lado, mostra uma diminuição no desempenho da taxa de transferência quando o número de solicitações simultâneas atinge 300 solicitações, conforme mostrado na tabela 3. Ao lidar com 1.000 solicitações, a implementação do PHP atinge apenas 48,70% da taxa de transferência. O NodeJS também mostra melhor desempenho no valor da taxa de transferência ao testar com o endpoint GET PARAMETER.

Tabela 3. Resultados da medição de throughput para terminal GET ALL

# de solicitações simultâneas	TAXA DE TRANSFERÊNCIA	
	PHP	NodeJS
1	100%	100%
100	100%	100%
200	100%	100%
300	90%	100%
400	93,25%	100%
500	90,20%	100%
600	85%	100%
700	71,43%	100%
800	57,63%	100%
900	50,11%	100%
1000	48,70%	100%

O resultado do experimento mostrou que a implementação do NodeJS tem melhor desempenho quando solicitada a lidar com várias solicitações simultâneas. Esse tipo de situação era um caso de uso comum no mundo real, no qual um serviço da Web deveria lidar com várias solicitações quase ao mesmo tempo. A arquitetura NodeJS com thread único de E/S sem bloqueio foi crucial para lidar com solicitações simultâneas, garantindo que todas as solicitações fossem atendidas no menor tempo possível.

6. Conclusão Em

nosso protótipo de implementação de API REST para sistema de informação acadêmico com duas tecnologias de servidor, exploramos o desempenho com dois parâmetros, tempo de resposta e throughput. Nosso protótipo com NodeJS e PHP com dois endpoints. A implementação do NodeJS tende a mostrar melhor desempenho para ambas as medições em ambos os endpoints. Outros experimentos precisam ser conduzidos para descobrir como a implementação NodeJS e PHP da API REST funciona sob uma configuração de banco de dados mais complexa e com terminais mais complexos, incluindo operações extensas de E/S.

Agradecimento Este
trabalho foi financiado pela Faculdade de Engenharia da Universitas Hasanuddin sob a concessão de 2019
Labo Based Education (LBE).

Referências

[1] Aswati S, Mulyani N, Siagian Y e Zikra Syah A 2015 Peranan Sistem Informasi dalam Perguruan Tinggi *Jurnal Teknologi dan Sistem Informasi* **1** 79-86.
[2] Barata, R., Silva, S., Martinho, D., Cruz, L., & Guerra e Silva, L 2014 Open APIs in
Sistemas de Informação para o Ensino Superior. *Umea.*
[3] Turner-McGrievy GM, Hales SB, Schoffman DE 2017 et al Escolhendo entre sites de design responsivo versus aplicativos móveis para sua intervenção comportamental móvel: apresentando quatro estudos de caso. *Transl Behav Med.* **7(2)** 224–232.
[4] Belqasmi, F., Singh, J., Melhem, SYB, Glitho, RH, 2012 Serviços da Web baseados em SOAP vs. RESTful: um estudo de caso para conferência multimídia *IEEE Internet Comput.* **16** 54–63.
[5] Choi M. 2012 Uma análise de desempenho do sistema de informações de API aberto RESTful *Tecnologia da informação de geração futura, notas de aula em ciência da computação* Springer Berlin Heidelberg 59–64.
[6] Posa R. 2015 Node JS Architecture – Single Threaded Event Loop Recuperado de https://www.journaldev.com/7462/node-js-architecture-single-threaded-event-loop.