Pemrosesan Data Sisi Klien Menggunakan Rakitan Web di Aplikasi Rendering Sisi Server

Rickard Elsen[1], Tryana Hadi Wijaya[2], Ridwan Setiawan[3]
Department of Computer Science
Institut Teknologi Garut
Garut, Indonesia
rickardelsen@itg.ac.id[1], 2106116@itg.ac.id[2], ridwansetiawan@itg.ac.id[3]

***Abstract*— Nowadays, web developers are required to choose between Single-Page Application (SPA) or Server-Side Rendering (SSR). Both types of web applications have their own advantages and disadvantages. Currently, most developers use Javascript to be able to perform computations on the client side even though the web application built is SSR. Scripts in Javascript are compiled by the web browser before the functions in the script can be used by the user. These functions will be executed on the client side so that computation does not need to be done on the server side. Since the use of Javascript is by sending source code to the client to be compiled and executed, WebAssembly (Wasm) allows the server to send those functions in precompiled binary files. The functions contained in the binary can be used by web applications and will be executed on the client side, so the browser doesn't need to compile it first to use existing functions. In this paper, we utilize Wasm to perform data processing on the client side of the SSR web application by performing some queries to preserved data.**

Keywords—WebAssembly, data processing, web technology, client side application

I. LATAR BELAKANG

WebAssembly (Wasm) memungkinkan browser web untuk mengakses fungsi eksternal dari biner yang telah dikompilasi, dalam perilaku yang sama seperti menggunakan Javascript. Wasm adalah format instruksi biner untuk mesin virtual berbasis tumpukan [1]. Wasm memungkinkan bahasa pemrograman yang mendukung Wasm, seperti Rust, C, dan C ++ [2], [3], untuk menggunakan Wasm sebagai target kompilasi dan menyebarkan biner yang dikompilasi ke aplikasi klien atau server. Wasm juga dirujuk akan bekerja sama dengan Web3, jika Ethereum Virtual Machine (EVM) dapat mendukung Wasm di masa depan [4]. Web3 adalah teknologi web terbaru yang memungkinkan aktivitas pengguna terdesentralisasi di internet [5]–[7].

Wasm memiliki empat fitur utama berdasarkan situs resminya::

1) Efficient and fast. Wasm tersedia di berbagai platform. Ini menggunakan mesin virtual berbasis tumpukan untuk mengeksekusi file biner Wasm sehingga dapat berjalan pada kecepatan asli dengan memanfaatkan kemampuan perangkat keras umum.

2) Safe. File biner Wasm akan dieksekusi di lingkungan kotak pasir sehingga tidak akan membahayakan sistem jika sesuatu gagal, atau tindakan penyalahgunaan telah dijalankan. Ini juga menerapkan kebijakan keamanan asal dan izin yang sama dari browser.

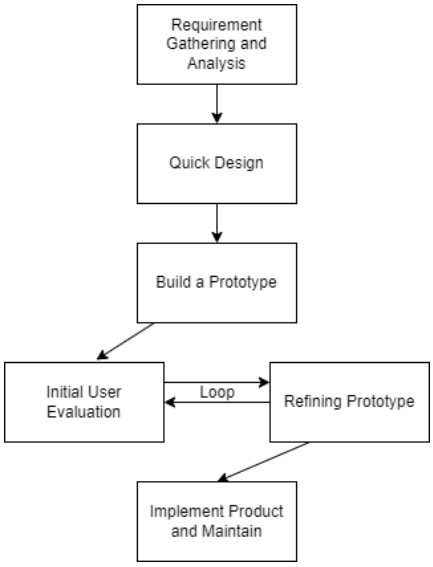
3) Open and debuggable. Wasm dapat dikompilasi ke file biner atau dapat diterjemahkan ke format WebAssembly Text (WAT). Format tekstual akan digunakan saat melihat sumber modul Wasm di web.

4) Part of the open web platform. Wasm dapat dipanggil di dalam Javascript atau juga dapat dipakai di luarnya. Wasm dapat diakses dengan perilaku yang sama seperti memanggil fungsi dari Javascript.

Terlepas dari semua manfaat di atas, Wasm masih memiliki beberapa masalah dalam keamanan dan bug [8]–[10]. Tapi itu wajar sebagai pembangunan di masa depan akan selalu berusaha memperbaiki masalah yang telah ditemukan. Wasm lebih unggul dalam hal speed[11], [12] Sehingga dapat digunakan untuk aplikasi yang mengutamakan performa.

II. METODE PENELITIAN

Pada penelitian ini, metodologi yang digunakan untuk menerapkan pengolahan data menggunakan Wasm dalam aplikasi web SSR adalah Prototyping Model [13]. Tahapan dalam metode ini adalah pengumpulan dan analisis kebutuhan, desain cepat, membangun prototipe, evaluasi pengguna awal, menyempurnakan prototipe, dan mengimplementasikan produk dan pemeliharaan, seperti yang terlihat pada Gambar 1.



Gambar 1 Fase model prototyping.

- Pengumpulan dan analisis kebutuhan. Tujuan dari fase ini adalah untuk mengumpulkan kebutuhan dan menganalisisnya. Karena aplikasi ini tidak memiliki pengguna, persyaratan akan ditentukan dengan merancang data dummy (data ini akan digunakan sebagai data yang diawetkan) dan perencanaan untuk membuat fungsi yang mungkin didasarkan pada data yang diawetkan.
- Desain cepat. Tujuan dari fase ini adalah untuk membuat diagram dari persyaratan yang dianalisis untuk menunjukkan bagaimana kueri bekerja pada data yang diawetkan. Desain hasil fase ini akan digambar menggunakan flowchart untuk menunjukkan alur proses query data.
- Membangun prototipe. Tujuan dari fase ini adalah untuk menerapkan kueri yang dirancang ke WebAssembly. Pada fase ini, kami membangun biner yang telah dikompilasi untuk digunakan dalam aplikasi web SSRInitial user evaluation. The purpose of this phase is to get user feedback about the prototype. But since this application has no users, in this phase we will test all the query to make sure all function works as defined in requirement.
- Menyempurnakan prototipe. Tujuan dari fase ini adalah untuk memperbaiki prototipe berdasarkan hasil tes dan memastikan semua persyaratan yang ditetapkan diterapkan pada perangkat lunak. Fase ini akan terus dilakukan hingga semua persyaratan dan fungsi berjalan dengan baik.
- Menerapkan produk dan memelihara. Tujuan dari fase ini adalah untuk menerapkan Wasm ke aplikasi web. Pada fase ini, kami memastikan biner yang telah dikompilasi sebelumnya dapat digunakan dalam aplikasi web SSR

III. HASIL DAN PEMBAHASAN

Dalam menerapkan metodologi, kami memutuskan untuk memadukan evaluasi pengguna awal, menyempurnakan prototipe, dan membangun fase prototipe karena tiga fase tersebut akan dilakukan berulang kali.

A. Pengumpulan dan Analisis Kebutuhan

Pada fase ini, kita membuat data dummy dengan lima field seperti yang terlihat pada Tabel 1. Kami menggunakan JSON Generator untuk menghasilkan sepuluh ribu data untuk digunakan sebagai data yang diawetkan. Data ini akan dikirim dari server ke klien dan akan diproses di sisi klien menggunakan Wasm.

Tabel 1 Model Data Yang Dipertahankan

No.	Field	Type	Value
1	ID	String	UUIDv4 without "-"
2	Name	String	Random name (first name and last name)
3	Age	Integer	Integer between 18 to 65
4	Email	String	Snack case name + random integer + "@itg.ac.id"
5	City	String	Random city

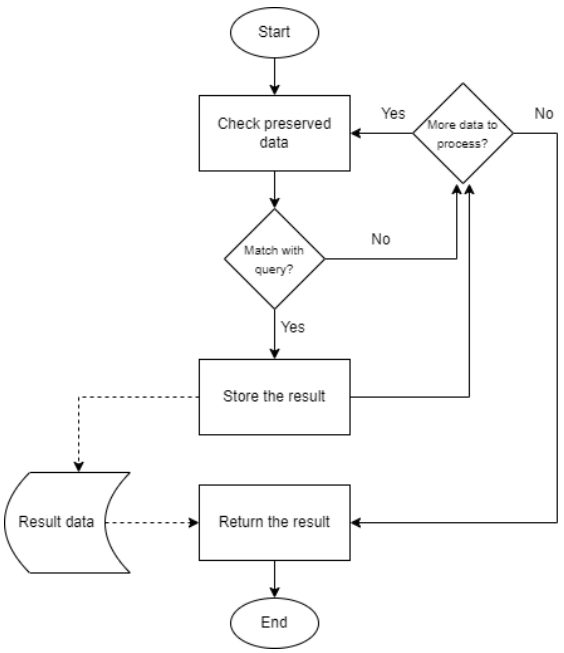
Berdasarkan data yang diawetkan, kami menentukan 4 fungsi yang akan diterapkan dalam Wasm seperti yang terlihat pada Tabel 2. Fungsi "search field by value" adalah fungsi dasar sehingga akan berlaku untuk semua bidang, tetapi kita akan menghitungnya sebagai satu fungsi karena perilaku yang sama. Fungsi "cari nama dengan nama depan atau nama belakang" juga menggunakan perilaku yang sama tetapi menggunakan data yang berbeda dari nama terpisah untuk memisahkan antara nama depan dan nama belakang.

Tabel 2 Daftar Fungsi

No.	Function	Field
1	Search field by value	All field
2	Search name by first name or last name	Name
3	Search age between	Age
4	Search multiple city	City

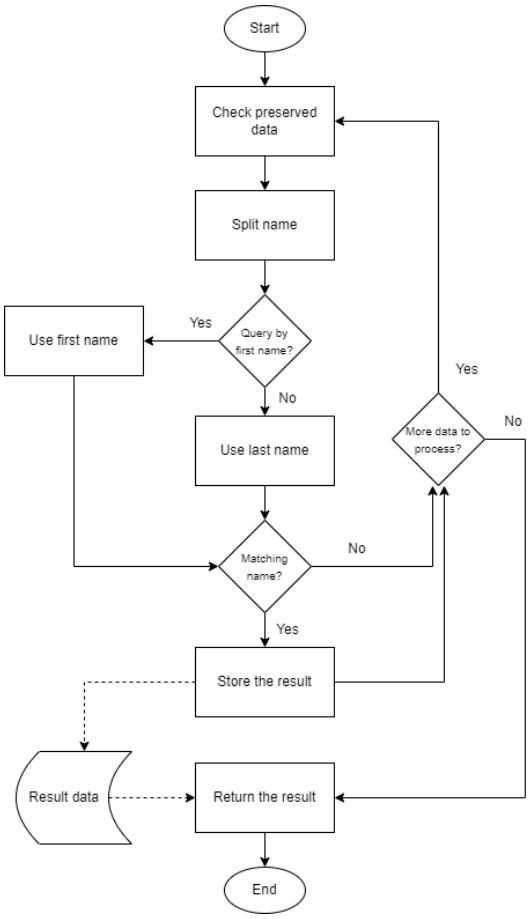
B. Quick Design

Pada fase ini, kami membuat desain untuk setiap fungsi untuk menjelaskan mekanisme bagaimana memproses data yang dipertahankan berdasarkan kueri yang diberikan. Kami menggunakan diagram alur untuk menunjukkan aliran pemrosesan data. Persyaratan awal untuk setiap proses adalah data yang dipertahankan telah diunduh dan disimpan dalam sistem klien sehingga biner Wasm dapat menggunakan data secara langsung.



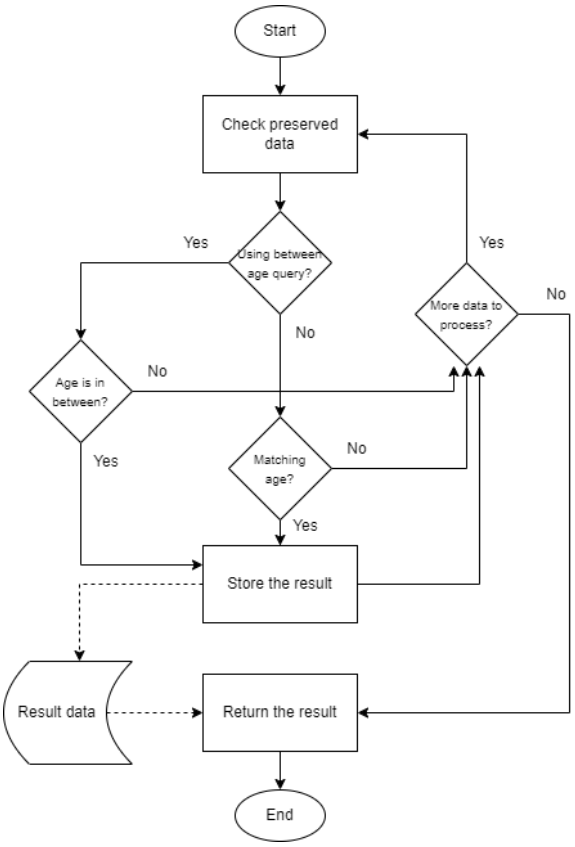
Gambar 2 Diagram alir pencarian berdasarkan fungsi bidang.

Seperti yang terlihat pada Gambar 2, mekanisme fungsi untuk memproses data berdasarkan nilai tergantung pada bidang adalah dengan mencocokkan data yang dipertahankan dengan kueri. Jika cocok, data akan disimpan ke data hasil. Proses ini akan diulang sampai semua data yang disimpan telah diperiksa dan dicocokkan dengan kueri.



Gambar 3 Diagram alur pencarian berdasarkan fungsi nama..

Seperti yang terlihat pada Gambar 3, mekanisme pencarian nama dengan nama depan atau nama belakang dimulai dengan memisahkan nama. Hasil dari nama split adalah nama depan dan nama belakang. Hasil ini akan digunakan tergantung pada kueri. Fungsi akan membagi nama dalam semua pengulangan untuk mencocokkan kueri dengan nama depan atau nama belakang. Hasilnya akan disimpan dan dikembalikan ketika semua data telah diperiksa.



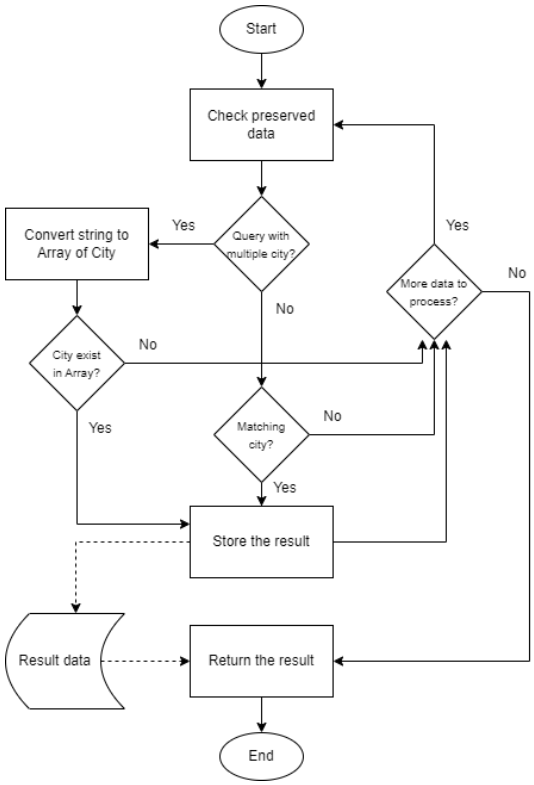
Gambar 4 Diagram alur pencarian berdasarkan usia antar fungsi

Seperti yang terlihat pada Gambar 4, mekanisme fungsi ini adalah untuk mencari data berdasarkan usia. Ini memiliki dua opsi: gunakan usia yang tepat atau gunakan usia antara dua angka. Sebagai contoh, jika kita mengirim kueri dengan nomor tunggal, fungsi ini akan mengembalikan data yang berisi usia dengan nilai yang sama dengan kueri yang diberikan. Jika kita mengirim dua angka, fungsi ini akan mengembalikan data yang berisi usia antara angka pertama dan angka kedua.

Seperti yang terlihat pada Gambar 5, mekanisme fungsi ini adalah untuk mencari data berdasarkan kota, dan dapat mencari berdasarkan beberapa kota. Kueri beberapa kota adalah dengan menambahkan koma untuk memisahkan antar kota. String akan dikonversi ke array kota dan data ini akan menjadi kueri dasar untuk menemukan data yang cocok. Dalam semua pengulangan, jika data pelestarian berisi kota di berbagai kota, data ini akan disimpan dalam data hasil dan akan dikembalikan ke aplikasi ketika semua data diperiksa.

C. Membangun Prototipe

Pada fase ini, kami menerapkan desain ke Wasm menggunakan AssemblyScript. AssemblyScript mengkompilasi varian TypeScript (superset JavaScript yang diketik) ke WebAssembly [14]. Kami menggunakan bahasa ini karena berbasis Javascript, jadi kami tidak mengalami kesulitan yang berarti saat mempelajarinya.



Gambar 5 Diagram alur pencarian beberapa fungsi kota

Karakteristik Wasm adalah mengekspor fungsi dalam format biner, sehingga mungkin tidak memiliki metode utama untuk dieksekusi. Fungsi yang diekspor ini dapat diakses menggunakan browser web dengan menggunakan modul WebAssembly [15]. Kami membuat lima fungsi untuk mengakomodasi semua kueri seperti yang terlihat pada Tabel 3.

Tabel 3 FUNGSI IMPLEMENTASI

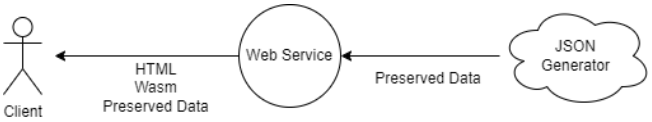
No.	Function Name	Parameters	Return Value
1	searchID	data: Object, query: Object	result: Object
2	searchName		
3	searchEmail		
4	searchAge		
5	searchCity		

Data dalam parameter adalah untuk meneruskan data yang diawetkan agar berfungsi. Untuk parameter dan nilai kembali, semua fungsi menerapkan skema yang sama. Hasil pengembangan akan dikompilasi ke biner Wasm menggunakan Assembly Script Compiler (ASC). Untuk aplikasi server, kami menggunakan NodeJS dan Express sebagai backend dan Nunjucks sebagai kerangka templateing. Kami menggunakan HTML minimalis sebagai UI web, Javascript dasar untuk memuat biner Wasm dan kueri pemformatan, dan Nunjucks untuk mengirim data yang diawetkan ke klien dengan memasukkan semua data dalam input HTML.

Untuk tujuan pengujian awal, kami membuat subset data yang dipertahankan yang berisi seratus data dan menerapkan kueri ke data ini. Kami membuat subset karena menggunakan semua data yang disimpan untuk pengujian sangat memengaruhi kinerja dan memengaruhi proses pengembangan. Dan sebagai hasil dari pengujian ini, ini menunjukkan bahwa semua fungsi berfungsi seperti yang diharapkan.

D. Penerapan Produk

Pada fase ini, kami berhasil mengatur semua komponen dan membuat layanan web. Seperti yang terlihat pada gambar 6, data yang disimpan disediakan oleh JSON Generator, dan kami menggunakan API mereka untuk memanfaatkan layanan mereka. Kami memutuskan untuk menghasilkan data baru setiap kali kami melakukan refresh halaman untuk membuktikan bahwa semua pemrosesan data dilakukan di sisi klien, karena data akan berubah jika pemrosesan data dilakukan di sisi server. Layanan Web akan mengirimkan data yang disimpan bersama halaman HTML (yang juga mencakup Javascript) dan biner Wasm. Dan di sisi klien, data yang diawetkan dapat diproses dengan kueri menggunakan fungsi dari biner Wasm.



Gambar 6 Arsitektur Layanan Web

Sebagai hasil dari fase ini, kami berhasil memanfaatkan Wasm untuk melakukan pemrosesan data di sisi klien. Pemuatan pertama aplikasi lambat, tetapi dapat dimengerti karena aplikasi memuat sepuluh ribu data. Tetapi setelah semua komponen halaman dimuat dan data yang diawetkan berhasil diunduh, proses kueri menjadi cepat. Kami tidak mengalami penundaan dalam melakukan kueri di semua fungsi. Data yang dipertahankan juga tidak berubah saat melakukan kueri terhadapnya. Ini membuktikan bahwa semua pemrosesan data dilakukan di sisi klien.

IV. KESIMPULAN

Kami berhasil memanfaatkan Wasm untuk melakukan pemrosesan data di sisi klien. Hasilnya berjalan seperti yang diharapkan. Wasm dapat memproses kueri data dengan kecepatan luar biasa. Tapi mungkin itu tergantung pada spesifikasi perangkat keras. Kami tidak melakukan pengujian untuk mengukur kecepatan pemrosesan antara Wasm dan Javascript. Namun ada beberapa artikel dan jurnal yang telah membahas perbandingan kecepatan antara Wasm dan Javascript. Dan sampai makalah ini selesai, kami menemukan bahwa Wasm lebih cepat daripada Javascript.

UCAPAN TERIMA KASIH

Penulis ingin mengucapkan terima kasih kepada Institut Teknologi Garut yang mendukung dan mendanai publikasi penelitian ini.

DAFTAR PUSTAKA

[1] "WebAssembly." https://webassembly.org/ (accessed Dec. 19, 2022).

[2] A. Hilbig, D. Lehmann, and M. Pradel, "An Empirical Study of Real-World WebAssembly Binaries," in *Proceedings of the Web*

Conference 2021, Apr. 2021, pp. 2696–2708. doi: 10.1145/3442381.3450138.

[3] K.-I. D. Kyriakou and N. D. Tselikas, "Complementing JavaScript in High-Performance Node.js and Web Applications with Rust and WebAssembly," *Electronics (Basel)*, vol. 11, no. 19, p. 3217, Oct. 2022, doi: 10.3390/electronics11193217.

[4] G. Zheng, L. Gao, L. Huang, and J. Guan, "WebAssembly(WASM)," in *Ethereum Smart Contract Development in Solidity*, 2021. doi: 10.1007/978-981-15-6218-1_11.

[5] F. A. Alabdulwahhab, "Web 3.0: The Decentralized Web Blockchain networks and Protocol Innovation," in *2018 1st International Conference on Computer Applications & Information Security (ICCAIS)*, Apr. 2018, pp. 1–4. doi: 10.1109/CAIS.2018.8441990.

[6] S. Aghaei, "Evolution of the World Wide Web : From Web 1.0 to Web 4.0," *International journal of Web & Semantic Technology*, vol. 3, no. 1, pp. 1–10, Jan. 2012, doi: 10.5121/ijwest.2012.3101.

[7] G. Korpal and D. Scott, "Decentralization and web3 technologies," 2022, [Online]. Available: /articles/preprint/Decentralization_and_web3_technologies/19727734/1

[8] D. Lehmann, J. Kinder, and M. Pradel, "Everything old is new again: Binary security of webassembly," *Proceedings of the 29th USENIX Security Symposium*, pp. 217–234, 2020.

[9] A. Romano, X. Liu, Y. Kwon, and W. Wang, "An Empirical Study of Bugs in WebAssembly Compilers," in *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, Nov. 2021, pp. 42–54. doi: 10.1109/ASE51524.2021.9678776.

[10] A. Romano and W. Wang, "WASim: Understanding WebAssembly Applications through Classification," *Proceedings - 2020 35th IEEE/ACM International Conference on Automated Software Engineering, ASE 2020*, pp. 1321–1325, 2020, doi: 10.1145/3324884.3415293.

[11] A. Haas *et al.*, "Bringing the web up to speed with WebAssembly," in *Proceedings of the 38th ACM SIGPLAN Conference on Programming Language Design and Implementation*, Jun. 2017, pp. 185–200. doi: 10.1145/3062341.3062363.

[12] A. Turner, "WebAssembly Is Fast: A Real-World Benchmark of WebAssembly vs. ES6," 2018. https://medium.com/@torch2424/webassembly-is-fast-a-real-world-benchmark-of-webassembly-vs-es6-d85a23f8e193 (accessed Dec. 19, 2022).

[13] R. Elsen, M. R. Nashrulloh, and A. Sutedi, "WALLET-BASED AUTHENTICATION ON COLLEGE INFORMATION SYSTEM," *Jurnal Teknik Informatika (JUTIF)*, vol. 3, no. 5, pp. 1373–1378, 2022, doi: https://doi.org/10.20884/1.jutif.2022.3.5.473.

[14] "https://www.assemblyscript.org/introduction.html."

[15] "https://developer.mozilla.org/en-US/docs/WebAssembly."