BAB II
TINJAUAN PUSTAKA

2.1 Penelitian Terdahulu

Penelitian terdahulu ialah penelitian yang dijadikan sebagai acuan pada penyelesaian tugas akhir di proyek atau penelitian yang sama. Penelitian terdahulu bisa menjadi panduan arahan perihal bagaimana melakukan penelitian terhadap topik tersebut wajib dilakukan serta pula dapat menyampaikan saran perihal bagaimana suatu perangkat lunak atau sistem menjadi lebih baik.

Penelitian sebelumnya telah dilakukan oleh Siti Saidah, Latipaturachmaniah, Rahul Syaban pada tahun 2023. Sistem ini dibangun bertujuan untuk mengimplementasikan arsitektur *microservices* pada aplikasi *point of sale* pada toko *flyover* stiker. Aplikasi tersebut dibangun menggunakan *framework* express js dan disisi *client* menggunakan react js.

Studi yang dilakukan oleh Regi Apriandi pada tahun 2023, dengan judul "Analisis Performa Aplikasi Berbasis Arsitektur Microservices pada Kubernetes Menggunakan Computer Cluster Raspberry PI 4", memiliki fokus untuk mengevaluasi kinerja perangkat lunak yang mengadopsi arsitektur *microservices*. Penelitian ini dilaksanakan dalam lingkungan Kubernetes yang diimplementasikan pada kluster Raspberry PI 4. Evaluasi kinerja tersebut didasarkan pada tiga aspek utama, yaitu *Quality of Service* (QOS), *Availability*, dan penggunaan sumber daya (*Resource utilization*).

Penelitain terdahulu selanjutnya ialah penelitian yang dilakukan oleh Lulu Agustin pada tahun 2023 dengan judul “Pengembangan *Microservices Core* pada

WEB Portal HiColleagues dengan *Framework* Echo Golang". Sistem ini berbasis *website* dengan *framework* echo golang. Penelitian ini dilakukan untuk mengembangkan *website* saat ini yang masih menggunakan arsitektur monolitik. Sehingga membagi menjadi beberapa *service* yang saling berinteraksi. *Service* yang akan di buat dengan menggunakan *framework* Echo ini karena SDM pada HiColleagues banyak yang berfokus pada bahasa pemrograman Golang.

Studi yang dilakukan oleh Umar Syarif pada tahun 2022, berjudul "Penerapan *Event-Driven Microservices* pada Aplikasi Layanan Penerimaan Peserta Didik Baru", bertujuan untuk mendemonstrasikan efektivitas implementasi *event-driven microservices* dalam konteks sistem Penerimaan Peserta Didik Baru (PPDB). Penelitian ini berusaha membuktikan bahwa pendekatan tersebut dapat menawarkan solusi yang efisien dan mudah diterapkan bagi lembaga pendidikan dalam mengelola proses pendaftaran siswa baru secara daring. Membagi sistem menjadi beberapa layanan kecil yang independen, menjadikannya lebih mudah dikelola dan di-skala. Teknologi Docker digunakan untuk menjalankan *container-container* aplikasi dalam lingkungan yang terisolasi.

Penelitain terdahulu selanjutnya ialah penelitian yang dilakukan oleh Mohammad Harry Khomas Saputra dan Luthfi Muhammad Nabil pada tahun 2021 dengan judul "Penerapan Arsitektur *Microservices* pada Sistem Tata Kelola Matakuliah Proyek Politeknik POS Indonesia". Sistem ini menggunakan *framework* express js untuk *backend*. Pengembangan baru dengan keaadan sistem monolitik saat ini menghadapi kesulitan karena harus mempelajari seluruh komponen sistem, sehingga penerapan aristektur *microservices* diperlukan.

Membagi menjadi beberapa bagian yang kecil meminimalisir kesalahan yang mengakibatkan mati total.

Berikutnya penelitian yang berjudul "Analisis pada Arsitektur *Microservices* Untuk Layanan Bisnis Toko *Online*".(Alchuluq & Nurzaman, 2021) menghasilkan perbandingan antara penerapan arsitektur *microservices* dengan arsitektur monolitik dengan merujuk pada hasil dari pengujian. Hasil dari pengujian yang dilakukan, menyimpulkan arsitektur *microservices* lebih unggul daripada arsitektur monolitik dengan spesifikasi yang sama.

Studi relevan berikutnya adalah karya Muhammad Fathan Radhiyan yang diselesaikan pada 2020, berjudul "Analisis dan Desain Arsitektur *Microservices* dengan GraphQL Sebagai API *Gateway* untuk Sistem Informasi Akademik AIS UIN Jakarta". Hasil dari penelitian ini berupa usulan rancangan arsitektur *microservices* yang memanfaatkan GraphQL sebagai API *Gateway*. Dalam proses perancangan, peneliti menerapkan teknik *decomposition by business capabilities* untuk menguraikan sistem menjadi komponen-komponen *microservices*. Hasil akhir sebatas analisa penerapan *microservices*.

Berikutnya penelitian yang berjudul "Implementasi Arsitektur *Microservices* pada Aplikasi WEB Pengajaran Agama Islam Home Pesantren".(Yuri Chandra Tri Putra, 2020) Penerapan arsitektur *microservice* dalam penelitian ini didukung oleh penggunaan RabbitMQ serta Docker guna kebutuhan *deployment*. Hasil yang didapat berasal penelitian ini adalah implementasi dari arsitektur monolitik ke arsitektur *microservices* sehingga bisa dilanjutkan pengembang selanjutnya tanpa harus mengubah semua komponen.

Studi relevan berikutnya adalah karya Mufrizal & Indarti yang dipublikasikan pada tahun 2019) dengan judul "Refactoring Arsitektur *Microservice* Pada Aplikasi Absensi PT. Graha Usaha Teknik". Penelitian ini bertujuan untuk mentransformasi sistem lama yang menggunakan arsitektur monolitik menjadi sistem baru berbasis arsitektur *microservice*. Dalam pengembangan perangkat lunak, para peneliti memanfaatkan Docker sebagai teknologi *containerization* dan menggunakan GitLab untuk memfasilitasi proses *deployment*.

Hasil penjelasan di atas dapat disimpulkan bahwa penelitan ini merupakan hasil adaptasi dari sepuluh penelitian sebelumnya. Dengan membandingkan metode yang belum pernah dilakukan pada penelitian terdahulu. Penelitian akan dilakukan kajian terhadap implementasi aristektur *microservice* pada *backend* sistem pembelian token listrik guna mencatat data transaksi secara *online*.

Tabel 2. 1 Penelitian Terdahulu

Nama Penerbit	Judul	Perbedaan
Siti Saidah, Latipaturachmaniah, Rahul Syaban (2023)	Implementasi Arsitektur *Micorservices* pada Aplikasi *Point of Sale* Toko *Flyover* Stiker	Dalam penelitian ini perbedaan ada pada *backend framework* yang digunakan. Pada penelitian yang dilakukan Siti Saidah, Latipaturachmaniah dan Rahul Syaban menggunakan expsress js dan pada penelitian ini sampai pada *end to end testing*. Sedangkan penulis menggunakan fastify dan tidak ada *end to end testing*.
Joseph Bagus Suryo Wicaksana (2023)	Implementasi Arsitektur Microservices dalam	Dalam penelitian ini perbedaan signifikan ada

	Pengembangan Sistem Manajemen Pemesanan Produk	pada *framework* yang digunakan dan tujuan utama aplikasi. *Framework* yang digunakan Joseph Bagus Suryo Wicaksana adalah express js sedangkan penulis menggunakan fastify js.
Regi Apriandi (2023)	Analisis Performa Aplikasi Berbasis Arsitektur *Microservices* pada Kubernetes Menggunakan *Computer Cluster* Raspberry PI 4	Dalam penelitian ini perbedaan signifikan pada hasil penelitian. Pada penelitian yang dilakukan Regi Apriandi hasil analisa performa aplikasi yang dibuat menggunakan Kubernetes *Computer Cluster* Raspberry PI 4. Sedangkan penulis aplikasi yang menerapkan *microservices*.
Lulu Agustin (2023)	Pengembangan *Microservices Core* pada WEB Portal HiColleagues dengan *Framework* Echo Golang	Dalam penelitian ini perbedaan signifikan ada pada penerapan bahasa pemograman. Penulis menggunakan javascript *framework* fastify sedangkan Lulu Agustin menggunakan Golang *framework* Echo.
Umar Syarif (2022)	Penerapan *Event-Driven Microservices* pada Aplikasi Layanan Penerimaan Peserta Didik Baru	Dalam penelitian ini perbedaan signifikan ada pada penerapan *framework* yang digunakan. *Framework* yang digunakan adalah *framework* Kraken menggunakan bahasa Go. *Deployment* yang digunakan adalah Docker.
Mohammad Harry Khomas Saputra, Luthfi Muhammad Nabil (2021)	Penerapan Arsitektur *Microservices* pada Sistem Tata Kelola Matakuliah Proyek Politeknik POS Indonesia	Dalam penelitian ini perbedaan signifikan ada pada *framework* yang digunakan dan tujuan utama aplikasi. *Framework* yang digunakan Mohammad Harry Khomas

		Saputra dan Luthfi Muhammad Nabil adalah express js sedangkan penulis menggunakan fastify js.
Liqoo Mumbahiz Alchuluq, Fahrul Nurzaman (2021)	Analisis pada Arsitektur *Microservices* Untuk Layanan Bisnis Toko *Online*	Dalam penelitian ini perbedaan signifikan pada hasil penelitian. Pada penelitian ini menghasilkan hasil uji coba antara arsitekutr *microservices* tanpa ada pengembangan atau penerapan pada aplikasi.
Muhammad Fathan Radhiyan (2020)	Analisis dan Desain Arsitektur *Microservices* dengan GraphQL Sebagai *API Gateway* untuk Sistem Informasi Akademik AIS UIN Jakarta	Dalam penelitian ini perbedaan signifikan pada hasil dan tujuan utama penelitian. Hasil dan tujuan dari penelitian ini hanya sebatas analisa dan desain aplikasi yang sudah berjalan.
Yuri Chandra Tri Putra (2020)	Implementasi Arsitektur *Microservices* pada Aplikasi WEB Pengajaran Agama Islam Home Pesantren	Dalam penelitian ini perbedaan signifikan pada penggunaan framework dan juga tujuan utama aplikasi.
Mufrizal & Indarti (2019)	*Refactoring* Arsitektur *Microservice* Pada Aplikasi Absensi PT. Graha Usaha Teknik	Dalam penelitian Mufrizal & Indarti perbedaan signifikan pada tujuan utama aplikasi. Tujuannya untuk efisiensi absensi sedangkan penulis untuk pembelian token listrik.

2.2 Teori Terkait

2.2.1 Microservices

Microservice terdiri dari komponen layanan yang kecil, terpisah, dan fokus pada tugas-tugas tertentu; mereka bekerja secara modular dan otonom, namun tetap terhubung satu sama lain melalui protokol. *Microservice* adalah pengembangan

lebih lanjut dari Arsitektur Layanan-*Oriented* karena terdiri dari berbagai proses yang berkomunikasi satu sama lain untuk membuat perangkat lunak yang kompleks menggunakan bahasa pengkodean antarmuka aplikasi (API) apapun. Arsitektur ini memungkinkan pencapaian tujuan utama dalam pengembangan perangkat lunak **(R. A. Putra, 2019)**.

2.2.2 Sistem

Sistem ialah suatu istilah yang sangat seringkali menjadi sebuah topik diskusi atau pembahasaan pada setiap karya ilmiah ataupun forum. Oleh sebab itu, penggunaan istilah atau kata yang sistematis bisa diterapkan dalam segala bidang dengan maknanya sendiri-sendiri. Secara umum, sistem juga dapat diartikan sebagai perpaduan komponen atau juga elemen yang saling terkait atau saling berhubungan di dalamnya, dimana sekumpulan elemen tersebut mampu saling berinteraksi satu sama lain untuk menggapai sebuah tujuan.

Menurut para pakar, sebuah sistem berarti piranti yang krusial serta sangat diperlukan oleh suatu perusahaan, instansi resmi atau yang lainnya. Sebab, dengan memiliki sistem yang sudah terintegrasi, suatu perusahaan atau instansi akan mempunyai kinerja yang lebih terarah, sistematis, dan efisien. Namun, agar bisa menghasilkan dampak positif dari penggunaannya, seluruh unsur-unsur yang berada di dalamnya harus saling bekerja sama untuk menggapai tujuan yang telah disepakati bersama.

Menurut Arifin, bersumber dari Webster New Collegiate Dictionary sistem berasal dari bahasa Yunani kata "*syn*" dan "*histanai*" dalam yang berarti satu. Jadi,

menurut Arifin Rahman, yang dimaksud dengan “sistem” yaitu sekumpulan dari beberapa pendapat (*a collection of opinion*), prinsip-prinsip yang menghasilkan suatu kesatuan dan saling terkait.

2.2.3 Basis Data

Basis data adalah sekumpulan informasi yang diorganisir secara sistematis dalam sistem komputer. Informasi ini mampu dimanipulasi atau dimodifikasi menggunakan aplikasi perangkat lunak tertentu. Peran basis data sangat krusial dalam pengelolaan informasi karena memungkinkan pengorganisasian data yang efisien, mencegah redundansi atau duplikasi data, dan meminimalkan risiko penyimpanan data yang tidak akurat. Selain itu, basis data memiliki karakteristik penting yang mencakup penentuan tipe data, pengaturan struktur, serta penetapan limitasi untuk data yang direncenakan untuk disimpan. Fitur-fitur ini memastikan integritas dan konsistensi data yang tersimpan dalam sistem.

Perangkat lunak yang dikenal sebagai sistem manajemen basis data (DBMS) esensial yang memfasilitasi penyimpanan dan pengambilan data dari media penyimpanan. DBMS berperan sebagai antarmuka yang memudahkan pengguna untuk mengelola, mengontrol, dan mengakses data secara efektif dan efisien.

Di antara berbagai jenis basis data relasional, beberapa yang paling dikenal dan banyak digunakan termasuk MariaDB, Microsoft SQL Server, dan Oracle. Namun, dalam konteks sistem pembelian token listrik yang dibahas, MySQL dipilih sebagai *platform* basis data yang digunakan.

MySQL, meskipun tidak disebutkan dalam daftar basis data paling populer, tetap menjadi pilihan yang layak dan efektif untuk sistem ini, menunjukkan bahwa pemilihan DBMS dapat disesuaikan dengan kebutuhan spesifik dari suatu proyek atau aplikasi.

MySQL adalah sistem manajemen basis data relasional yang populer, gratis, atau *open-source* (RDBMS) yang diciptakan, disebarluaskan, dan didukung oleh Oracle Corporation. Sistem relasional seperti MySQL menawarkan kemampuan untuk mengakses data dengan menggunakan bahasa pertanyaan terstruktur, yang juga dikenal sebagai SQL, dan menyimpan data dalam format tabel. Tentukan terlebih dahulu skema berbasis query MySQL sesuai dengan persyaratan, dan atur aturan untuk mengatur hubungan antar bidang dalam rekaman dan catatan. Melalui penggunaan gabungan, data MySQL dapat disimpan dalam tabel yang berbeda tetapi saling bergantung, mengurangi jumlah duplikasi data **(Chauhan, 2019)**.

2.2.4 *Unified Modeling Language* (UML)

Unified Modeling Language (UML) merupakan sebuah bahasa pemodelan yang berfungis untuk mendeskripsikan, memvisualisasikan, perancangan, dan pendokumentasian berbagai aspek sistem perangkat lunak, baik yang masih dalam tahap awal perencanaan maupun yang sedang dalam proses pengembangan. Definisi ini dikemukakan oleh Santoso & Nurmalina pada tahun (2017).

UML berfungsi sebagai metodologi perancangan untuk sistem berorientasi objek dan juga menjadi alat bantu dalam proses perancangan sistem. Dalam implementasinya, UML menyediakan berbagai jenis diagram, di antaranya

Sequence Diagram, Use Case Diagram, dan *Activity Diagram*. Masing-masing diagram ini memiliki peran penting sebagai instrumen dalam perancangan sistem.

A. Use Case Diagram

Menurut Nazir et al. (2022), "*Use Case Diagram* menggambarkan hubungan antara aktor dan *use case*, yang berguna untuk analisis dan desain sistem." *Use Case Diagram* menunjukkan tugas sistem yang harus dilakukan tanpa menunjukkan detail seperti struktur data, algoritma, atau lainnya. Pengguna yang menggunakan fungsi tertentu juga ditunjukkan pada diagram ini.

B. Sequence Diagram

Sequence Diagram menggambarkan interaksi antara objek dengan tujuan untuk memenuhi suatu tugas. *Sequence Diagram* berfokus pada pertukaran pesan secara struktur kronologis dari interaksi yang terjadi diantara pasangan komunikasi sehingga memungkinkan untuk berbagai macam bentuk interaksi agar dapat mengontrol struktur kronologis pertukaran pesan dan konsep dari modularisasi memungkinkan pengembang dalam pemodelan interaksi kompleks.

C. Activity Diagram

Activity Diagram adalah diagram yang mampu menggambarkan kontrol atau manipulasi dari mekanisme termasuk dari mekanisme alur penyebaran data yang menunjukkan sebuah tindakan sehingga dapat terlihat atau terdefinisikan sebagai aktifitas. *Activity Diagram* berfokus pada aspek proses model *procedural* dari sistem.

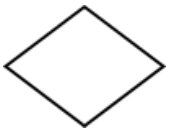
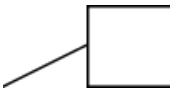
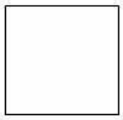
2.2.5 Flowchart

Dalam penyelesaian masalah, *flowchart* adalah cara untuk menunjukkan urutan logika **(Hanief & Jepriana, 2020)**. Dengan kata lain, *flowchart* adalah kumpulan langkah untuk menyelesaikan masalah yang ditunjukkan dengan simbol tertentu. *Flowchart* berfungsi sebagai dokumentasi program dan media komunikasi juga. Tujuannya adalah untuk memberikan penjelasan yang sederhana, rinci, dan sistematis tentang proses penyelesaian masalah.

Untuk mengatasi masalah yang muncul dalam suatu proses atau algoritma, diagram ini menawarkan penyelesaian langkah demi langkah. Berikut adalah simbol-simbol *flowchart* yang paling umum:

Tabel 2. 2 Simbol *Flowchart*

Gambar	Nama	Keterangan
	Garis Alir	Sebagai simbol menentukan arah algoritma dari satu proses ke proses lainnya.
	Terminal	Sebagai simbol awalan atau akhiran suatu proses.
	Proses / Langkah	Dalam diagram alir ini, simbol menunjukkan aktivitas yang akan dilakukan.
	Konektor Dalam Halaman	Penggunaan biasanya terjadi dalam situasi tertentu. Sama seperti tanda panah, berfungsi untuk menghubungkan proses ke proses lainnya. Bisa digunakan oleh lebih dari satu proses yang mengarah padanya, tetapi hanya dapat menghasilkan satu keluaran.
	Konektor Luar Halaman	Dalam beberapa situasi, diagram alir dapat mencakup lebih dari satu halaman. Oleh karena itu, simbol ini dapat digunakan untuk menghubungkan satu proses ke proses lain, yang merujuk ke halaman yang berbeda,

		seperti tanda panah, tetapi dengan tujuan yang berbeda.
	Titik Keputusan	Ketika suatu keputusan atau kondisi tertentu harus dipilih simbol ini selalu memiliki dua keluaran yang dapat digunakan untuk meneruskan aliran kondisi yang berbeda.
	Masukan / Keluaran	Hanya dapat digunakan dari urutan masukan ke keluaran, bukan sebaliknya, simbol ini dapat menunjukan data masuk dan keluar.
	Anotasi	Pada diagram alir, simbol ini menunjukkan komentar terhadap beberapa bagian, tetapi komentar tersebut tidak mempengaruhi proses yang sedang berjalan.
	*Predefined Process*	Untuk menunjukkan proses yang terlalu kompleks yang tidak dapat dijelaskan pada diagram alir, simbol ini merujuk pada diagram alir yang berbeda.
	Persiapan / Inisialisasi	Simbol ini digunakan untuk menggambarkan operasi yang tidak memiliki efek khusus selain mempersiapkan nilai untuk langkah atau proses berikutnya. Selain itu, simbol ini juga digunakan sebagai pengganti titik keputusan, terkadang berbentuk ketupat, ketika seseorang ingin menggunakan pengulangan pada suatu situasi.
	Kontrol/ Inspeksi	Selama inspeksi atau pengendalian, simbol ini menunjukkan langkah atau proses.

2.2.6 NodeJs

Node.js merupakan platform pengembangan yang menggunakan JavaScript sebagai bahasa pemrogramannya, dirancang untuk membangun aplikasi web. JavaScript, yang awalnya dikenal sebagai bahasa pemrograman untuk sisi klien atau *browser*, kini melalui Node.js dapat digunakan sebagai bahasa pemrograman sisi *server*, mirip dengan PHP, Ruby, dan Perl.

Keunggulan Node.js terletak pada kemampuannya untuk beroperasi di berbagai sistem operasi - Windows, Mac OS X, dan Linux - tanpa perlu memodifikasi kode programnya. Hal ini memberikan fleksibilitas yang signifikan bagi pengembang dalam mengembangkan dan menjalankan aplikasi berbasis web.

JavaScript membutuhkan engine yang cepat dan berkinerja tinggi untuk berfungsi sebagai bahasa pemrograman sisi *server*. Oleh karena itu, Node.js menggunakan Engine JavaScript Google bernama V8, yang juga digunakan oleh Chrome *browser*. Meskipun bahasa pemrograman sisi *server* biasanya bersifat *blocking*, Node.js memiliki sifat *non-blocking*, sama seperti JavaScript. *Blocking* di sini berarti menjalankan kode program hingga selesai sebelum beralih ke kode berikutnya. Sebagai ilustrasi, algoritma berikut:

1. Menerima permintaan halaman blog
2. Mengambil data blog dari *database*
3. Membuat html dari data blog
4. Mengirim respons ke klien

Dalam bahasa pemrograman yang membatasi penggunaan *multi-thread*, proses pengambilan data dari *database* dan penulisan data ke format HTML harus berjalan secara berurutan. Namun, Node.js menggunakan pendekatan *single-thread* yang berbeda. Alih-alih membuat *thread* untuk setiap tahapan proses, Node.js hanya membuat *thread* ketika ada *event* yang membutuhkannya.

Misalnya, saat mengambil data dari *database*, Node.js akan membuat *thread* untuk menulis data ke HTML hanya setelah data diterima, menggunakan

sistem *callback*. Pendekatan ini memungkinkan eksekusi proses tambahan yang tidak bergantung pada data sebelumnya tanpa harus menunggu data diambil selesai.

Metode ini memungkinkan Node.js untuk mengelola sumber daya dengan lebih efisien dibandingkan dengan pendekatan *multi-thread* tradisional, meskipun menggunakan CPU yang lebih lambat.

Joyent, sebuah perusahaan perangkat lunak dan infrastruktur *cloud*, mempekerjakan Ryan Dahl sebagai pengembang. Node.js, yang diciptakan dan ditunjukan oleh Dahl pada tahun 2009, membuka peluang baru bagi JavaScript. Inovasi ini memungkinkan JavaScript berfungsi sebagai bahasa pemrograman *server-side*, sejajar dengan bahasa-bahasa populer lainnya seperti PHP, ASP, C#, dan Ruby. Sebelum memutuskan menggunakan JavaScript, Dahl mempertimbangkan beberapa bahasa lain termasuk Haskell, Lua, dan C. Akhirnya, dia memilih JavaScript untuk Node.js karena ketertarikannya pada konsep *single-threaded* yang diterapkan di sisi *server*.

2.2.7 Javascript

JavaScript adalah bahasa pemrograman yang sangat diminati dan berkembang cepat. Sebagian besar peramban web terkenal mendukung JavaScript. Ini termasuk Google Chrome, Internet Explorer (IE), Mozilla Firefox, Netscape, dan Opera. JavaScript, bersama dengan HTML dan CSS, adalah salah satu teknologi inti *World Wide Web*, dan dapat dimasukkan ke halaman web menggunakan tag SCRIPT. JavaScript menciptakan halaman web lebih interaktif

dan salah satu bagian penting dari aplikasi web yang menarik. Parafrase dari kalimat tersebut:

JavaScript, yang awalnya terbatas pada penggunaan di browser web, kini telah berkembang pesat. Saat ini, mesin JavaScript dimanfaatkan dalam berbagai aplikasi di luar *browser*. Penggunaannya meliputi *server web* untuk pemrosesan di sisi *server*, sistem manajemen basis data, serta aplikasi *non-web* seperti perangkat lunak pengolah kata dan pembaca PDF. Selain itu, JavaScript juga berfungsi sebagai lingkungan *runtime* yang memungkinkan pengembangan aplikasi *mobile* dan *desktop*.

Brendan Eich dari Netscape pertama kali membuat JavaScript dengan nama Mocha, yang kemudian diubah menjadi LiveScript dan akhirnya dikenal sebagai JavaScript. Sebelum Navigator mendukung Java untuk membuat program non-Java lebih mudah digunakan, bahasa pemrograman bernama LiveScript kemudian dikembangkan dan dinamai JavaScript, meskipun tidak langsung terkait dengan Java. JavaScript adalah bahasa yang digunakan dalam pengembangan AJAX dan dapat digunakan untuk berbagai tujuan, seperti membuat efek *rollover* pada gambar dan teks.