**MERANCANG DAN MENGIMPLEMENTASIKAN PUSTAKA KOMPONEN AUTENTIKASI PENGGUNA YANG AMAN UNTUK *BLAZOR WASM* DAN *.NET CORE***

**SKRIPSI**

**Oleh**

**Isaiah Jamiel 2301934474**

**Izra Giffari 2301939166**

Diagram

Description automatically generated

**Cyber Security Program**

**Cyber Security Study Program**

**School of Computer Science**

**Universitas Bina Nusantara**

**Jakarta**

**2023**

**MERANCANG DAN MENGIMPLEMENTASIKAN PUSTAKA KOMPONEN AUTENTIKASI PENGGUNA YANG AMAN UNTUK *BLAZOR WASM* DAN *.NET CORE***

**SKRIPSI**

**diajukan sebagai salah satu syarat**

**untuk gelar kesarjanaan pada**

**Program Teknik Informatika**

**School of Computer Science**

**Jenjang Pendidikan Strata-1**

**Oleh**

**Isaiah Jamiel 2301934474**

**Izra Giffari 2301939166**

Diagram

Description automatically generated

**Cyber Security Program**

**Cyber Security Study Program**

**School of Computer Science**

**Universitas Bina Nusantara**

**Jakarta**

**2023**

**MERANCANG DAN MENGIMPLEMENTASIKAN PUSTAKA KOMPONEN AUTENTIKASI PENGGUNA YANG AMAN UNTUK *BLAZOR WASM* DAN *.NET CORE***

**SKRIPSI**

**Disusun oleh:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Isaiah Jamiel** |  | **Izra Giffari** |  |
| **2301934474** |  | **2301939166** |  |

**Disetujui Oleh:**

**Pembimbing**

**Nadia, S.Kom., M.TI**

**D5782**

**Yohan Muliono, S.Kom., M.Kom.**

***Head of Cyber Security Study program***

**Universitas Bina Nusantara**

**Jakarta**

**2023**

# BAB 1

**PENDAHULUAN**

## Latar Belakang

Peningkatan ketergantungan pada aplikasi perangkat lunak di berbagai sektor telah membuat autentikasi pengguna menjadi komponen kritis keamanan perangkat lunak. Memastikan bahwa hanya pengguna yang terautorisasi yang dapat mengakses sistem atau aplikasi sangat penting untuk melindungi data sensitif dan mencegah akses yang tidak sah.

Autentikasi pengguna adalah komponen kritis keamanan perangkat lunak, karena memastikan bahwa hanya pengguna yang terautorisasi yang dapat mengakses sistem atau aplikasi. Terdapat banyak pendekatan yang berbeda terhadap autentikasi pengguna, mulai dari kombinasi username dan password sederhana hingga metode yang lebih canggih seperti autentikasi biometrik dan autentikasi multi-faktor (Mittal & Gupta, 2018). Namun, menerapkan autentikasi pengguna yang aman dapat menjadi proses yang kompleks dan memakan waktu, terutama bagi para pengembang yang mungkin tidak memiliki keahlian di bidang keamanan.

Pustaka komponen menyediakan solusi yang berguna untuk masalah ini dengan menawarkan kode yang sudah dibangun, diuji, dan didokumentasikan yang dapat dengan mudah disatukan ke dalam proyek perangkat lunak. Dengan menggunakan pustaka komponen untuk autentikasi pengguna, para pengembang dapat fokus pada pembangunan fungsi inti aplikasi mereka, bukan mengeluarkan waktu dan sumber daya untuk mengembangkan dan menguji mekanisme autentikasi (Mittal & Gupta, 2018).

Pustaka komponen autentikasi digunakan dalam kerangka kerja untuk membantu memverifikasi identitas pengguna yang mengakses sumber daya atau melakukan tindakan tertentu dalam sebuah aplikasi. Pustaka-pustaka ini biasanya menyediakan mekanisme untuk membuat dan mengelola akun pengguna, serta melakukan verifikasi kredensial pengguna antara lain kombinasi nama pengguna dan kata sandi.

Ada beberapa tujuan dalam penggunaan library autentikasi:

1. Untuk memastikan bahwa hanya pengguna yang terautorisasi yang dapat mengakses sumber daya yang sensitif atau terbatas.
2. Untuk memberikan cara yang konsisten dan aman untuk mengelola akun pengguna dan kata sandi di seluruh aplikasi.
3. Untuk mengurangi beban kerja developer dengan menyediakan set fungsi autentikasi yang sudah dibangun sebelumnya yang dapat dengan mudah diintegrasikan ke dalam sebuah aplikasi.
4. Untuk memberikan cara standar untuk menangani tugas autentikasi umum, seperti pemulihan kata sandi.
5. Secara keseluruhan, penggunaan library autentikasi dalam kerangka kerja membantu meningkatkan keamanan dan kemudahan penggunaan sebuah aplikasi, sambil juga memudahkan bagi developer untuk mengimplementasikan fungsi autentikasi.

Blazor telah memperoleh minat dan adopsi yang signifikan dalam komunitas developer sejak dirilis pada tahun 2018. Menurut Survei Pengembang Stack Overflow untuk tahun 2021, Blazor menduduki peringkat ke-13 kerangka kerja web paling populer di kalangan developer, yang merupakan peningkatan signifikan dari peringkat pada tahun-tahun sebelumnya.

Ada beberapa alasan mengapa Blazor mendapatkan popularitas dalam beberapa tahun terakhir:

1. Peningkatan produktivitas pengembang: Blazor memungkinkan pengembang menggunakan C# dan .NET untuk membangun aplikasi web statis sisi klien, yang bisa lebih produktif bagi pengembang yang sudah terbiasa dengan teknologi ini.
2. Basis kode bersama: Karena Blazor menggunakan CLR yang sama dengan ekosistem .NET, Blazor memungkinkan pengembang untuk berbagi kode antara sisi server dan klien dari suatu aplikasi, yang dapat meningkatkan pemeliharaan basis kode dan mengurangi duplikasi upaya.
3. Mengurangi ketergantungan pada JavaScript: Blazor memungkinkan pengembang membuat aplikasi web tanpa terlalu bergantung pada JavaScript, yang dapat bermanfaat bagi pengembang yang lebih suka bekerja dengan bahasa C# dan bahasa yang kompatibel dengan .NET lainnya. Blazor pun tetap memberikan opsi untuk berinteraksi dengan Javascript untuk mendukung fitur yang hanya ada di Javascript.
4. Performa yang ditingkatkan: Aplikasi Blazor akan di kompilasi ke WebAssembly, maka akan lebih cepat dibanding aplikasi berbasis JavaScript, terutama untuk aplikasi yang sering memerlukan pembaruan UI.

Secara keseluruhan, Blazor menawarkan sejumlah manfaat bagi pengembang dan organisasi, dan popularitasnya kemungkinan besar akan terus tumbuh karena semakin banyak orang yang menyadari kemampuannya.

Dengan naiknya popularitas Blazor dalam dunia web, kami melihat bahwa minim adanya library yang dapat membantu dalam menjaga keamanan website berbasis Blazor dan hal ini menyebabkan beberapa masalah autentikasi potensial yang mungkin dihadapi oleh para pengembang ketika bekerja dengan framework Blazor. Beberapa masalah umum termasuk:

* Mengimplementasikan autentikasi: Tergantung pada kebutuhan spesifik aplikasi, mengimplementasikan autentikasi dalam aplikasi Blazor dapat menjadi tantangan. Para pengembang mungkin perlu mengintegrasikan dengan penyedia autentikasi eksternal, seperti OAuth atau OpenID Connect, atau membangun solusi autentikasi kustom mereka sendiri.
* Mengelola sesi pengguna: Aplikasi Blazor menggunakan model client-side, yang berarti sesi pengguna harus dikelola di sisi client. Ini dapat lebih kompleks daripada mengelola sesi pengguna di sisi server, dan para pengembang mungkin perlu mempertimbangkan masalah seperti kadaluarsa sesi dan refresh.
* Menjamin keamanan data: Ketika bekerja dengan data sensitif, penting untuk memastikan bahwa data tersebut dikirim dan disimpan dengan aman. Ini dapat menjadi tantangan dalam aplikasi Blazor, karena data sering dikirim melalui jaringan dalam teks biasa. Para pengembang mungkin perlu mempertimbangkan implementasi enkripsi atau tindakan keamanan lain untuk melindungi data sensitif.
* Menangani kesalahan: Jika terjadi kesalahan selama proses autentikasi, sulit bagi para pengembang untuk men-debug dan mencari tahu masalah tersebut. Ini dapat menjadi terutama sulit jika kesalahan terjadi di sisi client aplikasi.

Mengingat juga banyaknya kasus kebocoran data-data di Indonesia dalam beberapa tahun terakhir ini yang memungkinkan data yang bocor adalah data terkait kredensial login. Maka dari itu, kami melihat bahwa perancangan library komponen autentikasi akan berdampak positif dalam dunia perkembangan web yang terus berkembang.

Tujuan dari tesis ini adalah merancang dan melakukan implementasi pustaka komponen autentikasi pengguna yang aman untuk digunakan dengan Blazor WebAssembly (WASM) dan .NET Core. Pustaka tersebut akan memberikan para pengembang dengan berbagai metode autentikasi yang dapat dengan mudah disatukan ke dalam aplikasi mereka, yang membantu untuk memastikan bahwa autentikasi pengguna diimplementasikan secara konsisten dan aman. Pustaka akan diuji dan dievaluasi untuk memastikan bahwa sesuai dengan standar keamanan yang dibutuhkan dan mudah digunakan oleh para pengembang.

## Rumusan Masalah

* + 1. Bagaimana cara membuat sebuah library komponen autentikasi yang modular dan mudah di implementasikan untuk para penggunanya?
    2. Bagaimana cara mengimplementasikan pustaka komponen autentikasi pengguna yang aman menggunakan Blazor WebAssembly dan .NET Core?
    3. Bagaimana cara menjalankan seluruh proses autentikasi dengan cepat namun tetap aman?

## Ruang Lingkup

Berikut ini adalah ruang lingkup penelitian yang dilakukan:

* Melakukan tinjauan literatur mengenai metode dan praktik terbaru dalam otentikasi pengguna, keamanan aplikasi web, serta penggunaan Blazor WASM dan .NET Core.
* Merancang pustaka komponen otentikasi pengguna yang aman dengan mempertimbangkan arsitektur dan fitur yang dibutuhkan dalam konteks Blazor WASM dan .NET Core.
* Melakukan pengujian terhadap pustaka komponen otentikasi untuk memastikan keamanan, kinerja, dan fungsionalitasnya. Mengevaluasi keberhasilan implementasi pustaka dalam memenuhi kebutuhan otentikasi pengguna.

## Tujuan dan Manfaat

Tujuan dilakukannya penelitian ini adalah sebagai berikut:

* + - 1. Memberikan solusi untuk developer dalam membangun situs berbasis Blazor dan .NET Core yang aman dengan mudah dan cepat secara pengmbangan.
      2. Meningkatkan kualitas keamanan website yang menggunakan kerangka kerja blazor dan .NET.

Adapun manfaat dari penelitian ini adalah:

* + - 1. Mempermudah developer dalam mengembangkan situs dengan sistem autentikasi yang aman.
      2. Menyediakan solusi autentikasi untuk developer .NET yang bisa dipercaya

## Metodologi

Penelitian yang dilakukan terdiri dari beberapa tahap serta metode yang digunakan, yaitu:

* + 1. **Metode Pengumpulan Data**

1. Interview

Teknik ini dilakukan dengan melakukan interview narasumber yang berkaitan dengan pustaka komponen dan pengembangan aplikasi berbasis web dengan tujuan memvalidasi masalah dan kebutuhan serta manfaat yang ditawarkan oleh komponen pustaka yang akan dibuat.

1. Perbandingan aplikasi sejenis

Teknik ini dilakukan untuk mengetahui fitur-fitur yang sudah ada maupun belum dari pustaka komponen yang sejenis. Hal ini dilakukan sebagai bahan pertimbangan untuk melihat perbedaan yang dapat dijadikan sebagau fungsi tambahan dan nilai pada pustaka komponen yang akan dibuat.

* + 1. **Metode Pengembangan**

Metode pengembangan aplikasi yang diterapkan dalam penelitian ini adalah *Waterfall Model*. *Waterfall Model* adalah suatu pendekatan dalam rekayasa perangkat lunak yang digunakan untuk mengembangkan aplikasi. Metode ini melibatkan serangkaian tahapan yang dimulai dengan analisis masalah dan pengumpulan kebutuhan pengguna, perencanaan jadwal dan pembagian tugas selama pengembangan aplikasi, desain aplikasi menggunakan UML, pembangunan aplikasi sesuai dengan desain yang telah dibuat, dan akhirnya melakukan evaluasi terhadap aplikasi yang telah dibangun.

Penulis memilih menggunakan *Waterfall Model* karena model ini dapat memberikan arahan yang jelas dalam pembuatan aplikasi. Dengan adanya requirement dan kebutuhan pengguna yang sudah terdefinisi sejak awal, model ini dapat membantu menghindari kemungkinan perubahan kebutuhan yang muncul ketika proses perancangan aplikasi sudah dimulai. Dengan demikian, penggunaan *Waterfall Model* diharapkan dapat memastikan pengembangan aplikasi berjalan dengan terstruktur dan terarah.

## Sistematika Penulisan

Penulisan Tugas Akhir ini dibagi menjadi 5 (lima) bab yaitu:

BAB 1 Pendahuluan

Bab ini berisi tentang latar belakang yang menjadi dasar dilakukannya penelitian, rumusan masalah, ruang lingkup, tujuan dan manfaat, metode penelitian yang digunakan, serta sistematika penulisan.

BAB 2 Tinjauan Pustaka

Dalam bab ini, akan dijelaskan landasan teori yang menjadi dasar panduan dalam pembuatan aplikasi ini, serta metode yang digunakan dalam penelitian ini.

BAB 3 Metode Penelitian

Bab ini berisi kerangka berpikir, metode penelitian dalam menganalisa masalah, perencanaan dalam membuat aplikasi, pengembangan algoritma, pengembangan pustaka komponen, perancangan sistem dengan UML, dan metode yang digunakan dalam membuat pustaka komponen.

BAB 4 Hasil dan Pembahasan

Bab ini berisi spesifikasi sistem, hasil pengembangan aplikasi secara keseluruhan, serta hasil uji coba dari metode yang digunakan. Metode yang digunakan dikaitkan dengan teori dan konsep dari literatur terkait. Bab ini juga berisi hasil evaluasi yang meliputi evaluasi sistem, dan evaluasi aplikasi secara keseluruhan.

BAB 5 Simpulan dan Saran

Bab ini berisi simpulan dari seluruh pembahasan dan menjawab rumusan masalah yang dipaparkan pada bab 1. Bab ini juga berisi saran dan usulan untuk pengembangan aplikasi yang lebih baik di waktu mendatang.

# BAB 2

**TINJAUAN PUSTAKA**

### **Pustaka Komponen**

Komponen pustaka adalah kumpulan komponen perangkat lunak yang dapat digunakan kembali yang dapat digunakan untuk membangun atau meningkatkan fungsi aplikasi perangkat lunak. Komponen-komponen ini biasanya dirancang untuk modul dan independen, sehingga dapat dengan mudah disatukan ke dalam berbagai sistem yang berbeda. (newtarget, 2022)

Pustaka komponen dapat mencakup berbagai jenis komponen yang berbeda, seperti elemen antarmuka pengguna, modul logika bisnis, komponen akses data, dan lainnya. Mereka mungkin dikembangkan dan dipelihara oleh satu organisasi, atau mungkin dikumpulkan dari berbagai sumber dan tersedia untuk digunakan oleh komunitas yang lebih besar. (newtarget, 2022)

### **Autentikasi**

Autentikasi adalah proses verifikasi identitas pengguna, perangkat, atau sistem (Jain, 2014). Ini merupakan aspek penting keamanan komputer yang memastikan hanya individu atau entitas yang terautorisasi yang dapat mengakses sumber daya atau melakukan tindakan pada sistem (Barker et al., 2018).

Terdapat beberapa metode autentikasi yang dapat digunakan untuk memverifikasi identitas pengguna atau sistem. Metode-metode tersebut antara lain:

* + 1. **Autentikasi Berbasis Kata Sandi**

Autentikasi berbasis sandi adalah suatu proses yang digunakan untuk memverifikasi identitas seseorang atau sebuah sistem dengan menggunakan sebuah kombinasi dari nama pengguna dan sandi (password). Sandi biasanya merupakan sebuah string (baris teks) yang diketahui hanya oleh pengguna yang bersangkutan, dan digunakan sebagai faktor autentikasi utama.

* + 1. **Autentikasi Dua Faktor**

Autentikasi dua faktor (two-factor authentication atau 2FA) adalah suatu proses yang digunakan untuk memverifikasi identitas seseorang atau sebuah sistem dengan menggunakan dua faktor yang berbeda. Dua faktor tersebut adalah faktor yang diketahui hanya oleh pengguna (seperti sandi atau kata sandi) dan faktor yang tidak diketahui oleh pengguna (seperti kode verifikasi yang dikirim ke nomor telepon atau alamat email pengguna).

* + 1. **Autentikasi Multifaktor**

Autentikasi faktor ganda (MFA) adalah proses keamanan yang memerlukan lebih dari satu metode otentikasi dari kategori credential yang independen untuk memverifikasi identitas pengguna. Ini berarti bahwa selain password, faktor kedua diperlukan untuk l*ogin* ke sistem atau mengakses informasi yang dilindungi. Tujuan MFA adalah untuk membuat lebih sulit bagi pengguna yang tidak sah untuk mengakses sistem atau data, bahkan jika mereka telah mendapatkan nama pengguna dan sandi yang valid.

### **Enkripsi dan Dekripsi**

Enkripsi adalah proses penting dalam ilmu komputer dan keamanan siber yang melibatkan transformasi teks biasa atau data tak terenkripsi menjadi bentuk yang tidak dapat dibaca oleh siapa pun tanpa kunci dekripsi. Data yang terenkripsi disebut sebagai teks sandi. Enkripsi digunakan untuk melindungi informasi sensitif dari akses oleh individu seperti hacker atau kriminal.

Salah satu algoritma enkripsi tertua dan paling banyak digunakan adalah Advanced Encryption Standard (AES). AES adalah algoritma enkripsi simetris yang beroperasi pada blok data dengan ukuran tetap. Algoritma ini telah banyak diadopsi oleh organisasi dan pemerintah di seluruh dunia karena memiliki keamanan yang kuat (Rijmen, Daemen, & Preneel, 2002).

Algoritma enkripsi penting lainnya adalah algoritma RSA, yang didasarkan pada kompleksitas matematika faktorisasi bilangan besar. RSA adalah algoritma enkripsi asimetris yang menggunakan kunci publik untuk enkripsi dan kunci privat untuk dekripsi. Algoritma ini telah banyak digunakan dalam mengamankan komunikasi, tanda tangan digital, dan pertukaran kunci (Rivest, Shamir, & Adleman, 1978).

Selain algoritma enkripsi dasar ini, terdapat kemajuan dalam teknik enkripsi seperti kriptografi kurva eliptik (ECC), enkripsi homomorfik, dan kriptografi pasca-kuantum. ECC dikenal karena kemampuannya untuk memberikan tingkat keamanan yang sama dengan algoritma enkripsi tradisional dengan ukuran kunci yang lebih kecil, sehingga cocok untuk perangkat yang terbatas sumber daya (Lopez & Dahab, 1999). Enkripsi homomorfik memungkinkan komputasi dilakukan pada data yang terenkripsi tanpa dekripsi, yang sangat berguna dalam skenario di mana privasi merupakan perhatian utama (Gentry, 2009). Kriptografi pasca-kuantum berfokus pada pengembangan algoritma enkripsi yang tahan terhadap serangan oleh komputer kuantum, yang memiliki potensi untuk merusak banyak algoritma enkripsi yang ada (Buchmann & Dahmen, 2011).

## C#

C# adalah bahasa pemograman berorientasi pada obyek yang modern, inovatif, *open-source*, dan *cross-platform* (Microsoft, 2022). C# memiliki sistem tipe data yang kuat, sehingga setiap variabel dan konstanta harus memiliki tipe data (Microsoft, 2022). Meskipun C# merupakan bahasa yang berorientasi pada obyek pada akarnya, C# juga menawarkan paradigma lain untuk developer bisa menggunakan bahasa tersebut. Antara lain, C# memungkinkan developer untuk mengambil pendekatan fungsional dan bisa mengimplementasikan lambda (BillWagner, GeWarren, pkulikov, wuyuansushen, & DCtheGeek, 2022).

C# awalnya dibuat untuk menjadi pesaing bahasa Java*.* Pada versi-versi awal C#, bahasa buatan Microsoft ini sangat mirip dengan Java. Namun, semakin berkembangnya C#. C# semakin memiliki jati diri nya sendiri sebagai bahasa pemograman. C# mencoba untuk menghilangkan *boilerplate* dari struktur kodenya dan mulai menawarkan fitur-fitur yang mengarah kepada paradigma pemograman lain.

Salah satu fitur yang paling digemari oleh pengguna C# adalah LINQ. LINQ menjadi salah satu nilai jual paling tinggi dari C# atau bahasa lain di framework .NET. LINQ memungkinkan developer untuk mengolah data secara deklaratif. Dimana bahasa lain banyak yang belum memiliki fitur ini.

List<String> a;

a.Where(a => a == "test");

cth. LINQ

Selain LINQ, C# dan bahasa-bahasa lain di framework .NET juga memiliki sebuah IDE atau *Integrated development environment* yang para developer terutama di bidang korporat bisa percaya akan terus ada dan di support oleh *Microsoft* sendiri. IDE ini adalah Visual Studio. Visual studio memiliki banyak fitur, yang dimana pada masa Visual Studio baru di terbitkan ia merupakan IDE terbaik yang ada. Fitur-fitur seperti *debug,* pelengkap kode secara otomatis, dan juga fitur-fitur lain ini dinilai oleh perusahaan bisa membuat pemograman lebih cepat, sehingga bisa menghemat biaya.

## Framework

## .NET

.NET (dibaca dotnet) merupakan sebuah platform developer yang gratis, *cross-platform*, dan *open-source.* Aplikasi .NET dapat ditulis dalam 3 bahasa yang didukung secara resmi, yaitu C#, F#, dan Visual Basic. .NET memiliki API yang konsisten, sehingga aplikasi yang berbeda, jika berbasis .NET masih bisa berbagi kode dan menggunakan library yang sama (Microsoft, 2022).

.NET sebuah microsoft pertama kali dibuat untuk menyaingi Java dengan teknologi JVM nya. Untuk menyaingi *Java Virtual Machine* (disingkat JVM) milik Java, .NET memiliki teknologi nya sendiri yang dinamakan *Common Language Runtime* (disingkat CLR). CLR dibuat demi mempermudah programmer untuk mencapai beberapa hal, antara lain manajemen memori dan manajemen *thread*. CLR ini merupakan sebuah lapisan penghubung antara bahasa pemograman dan Sistem Operasi.

Seperti namanya, CLR dibuat untuk menyatukan lebih dari satu bahasa pemograman agar bisa berjalan dan berinteraksi antara satu sama lain melalui dua teknologi, *Common Language Specification* (disingkat CLS) dan *Common Type System* (disingkat CTS).

*Common language specification* merupakan seperangkat aturan dan larangan yang berfungsi untuk menjembatani antar bahasa pemograman. Hal ini memungkinkan untuk lebih dari satu bahasa digunakan di satu proyek yang sama untuk tidak saling merusak satu sama lain. CLS juga merupakan teknologi yang memungkinkan sebuah class di sebuah bahasa menerapkan *inheritance* sebuah class dari bahasa lain. Seperti, sebuah *class* di C# bisa melakukannya terhadap sebuah class di VB.NET. Namun, hal ini tidak sempurna dan memiliki larangan-larangan tertentu. Seperti, sebuah *class* di C++ tidak bisa di *inherit* oleh class di C#. Contoh lain adalah, saat memanggil *method* dari bahasa lain. Bahasa yang memanggilnya harus mengikuti aturan dari bahasa yang dipanggil. Salah satu contoh nya adalah, saat memanggil bagian-bagian dari kode di C#, VB.NET tetap harus mengikuti aturan di C#. Jadi, saat pemanggilan bagian dari C#, menjadi *case sensitive*.

Di sisi lain, *Common Type System* merupakan seperangkat aturan yang menyatukan tipe-tipe data yang digunakan di berbagai bahasa pemograman. CTS akan mengetahui bahwa int di C# adalah sama dengan integer di VB.NET. CTS akan menyatakan bahwa mereka kompatibel sehingga pemanggilan data bisa dilakukan dengan benar.

Kedua CTS dan CLS digunakan di dalam CLR untuk menginterpretasi kode yang telah di kompilasi. Saat ini, .NET memiliki 3 bahasa pemograman utama yang di dukung secara penuh, yakni C#, VB.NET, dan F#. Blazor WASM

## Blazor

Blazor adalah kerangka kerja berbasis .NET yang menggunakan teknologi *WebAssembly* untuk memungkinkan membuat sebuah situs statis sisi klien responsive tanpa menggunakan JavaScript. Meskipun demikian, Blazor tetap dapat memanggil library dan API Javascript melalui C# (Microsoft, 2022).

Salah satu keunggulan utama Blazor adalah kemampuannya untuk membuat aplikasi web yang sepenuhnya berbasis komponen. Dalam Blazor, pengembang dapat membuat komponen UI yang dapat digunakan kembali dan dirangkai bersama untuk membentuk aplikasi yang lebih besar. Setiap komponen memiliki logika dan tampilan sendiri, dan dapat berinteraksi dengan komponen lain dalam aplikasi. Ini memungkinkan pengembang untuk membagi aplikasi menjadi bagian-bagian yang lebih kecil, yang dapat dikelola dan diuji secara terpisah.

Blazor mendukung dua mode utama: Blazor Server dan Blazor WebAssembly. Pada mode Blazor Server, logika aplikasi dijalankan di server dan UI diperbarui secara real-time melalui koneksi jaringan ke browser pengguna. Ini memungkinkan pengembang untuk membuat aplikasi web yang responsif dengan interaksi yang halus, tanpa harus mengirimkan kode C# ke browser. Di sisi lain, Blazor WebAssembly memungkinkan pengembang untuk menjalankan kode C# langsung di browser pengguna melalui teknologi WebAssembly. Ini memberikan pengalaman pengguna yang lebih cepat dan lebih kaya, karena kode aplikasi dieksekusi di sisi klien.

Blazor juga mendukung integrasi yang baik dengan ekosistem .NET yang ada. Pengembang dapat menggunakan banyak library dan alat yang sudah ada dalam ekosistem .NET untuk mempercepat proses pengembangan. Selain itu, Blazor juga mendukung penggunaan komponen UI dari Bootstrap dan Material Design, serta integrasi dengan layanan cloud seperti Azure.

Dengan fitur-fitur yang kuat dan kemampuan untuk menghindari ketergantungan pada JavaScript, Blazor menawarkan pendekatan yang menarik untuk pengembangan aplikasi web. Dengan menggunakan C# dan .NET, pengembang dapat memanfaatkan kekuatan dan produktivitas bahasa dan kerangka kerja yang sudah mapan, sambil tetap menghasilkan aplikasi web yang modern dan responsif.

## Prototyping

## Github

## Privasi dan Proteksi Data Pengguna

## Security Testing

## SDLC (Software Development Life Cycle)

# BAB 3

**METODE PENILITIAN**

* 1. **Kerangka Berpikir**

Dalam proses pengembangan aplikasi untuk skripsi ini, penulis menggunakan metode SDLC Waterfall untuk memastikan alur pengembangan yang terstruktur. Dengan menggunakan metode SDLC Waterfall, penulis dapat mengelola proses pengembangan secara sistematis. Lingkup dari aplikasi yang sedang dikembangkan tidak terlalu besar, sehingga mengurangi kemungkinan perubahan yang signifikan terhadap kebutuhan aplikasi saat memasuki tahap pengembangan.

A picture containing text, screenshot, font, design

Description automatically generated

Pada metode SDLC Waterfall terdapat 6 tahapan yang digunakan penulis dalam mengembangkan komponen pustaka ini:

1. *Analisis Kebutuhan*: Mengumpulkan dan menganalisis kebutuhan untuk komponen perpustakaan. Memahami fungsi, *user interface*, harapan kinerja, dan spesifikasi lainnya.
2. *System Design*: Membuat desain terperinci untuk komponen perpustakaan. Mendefinisikan arsitektur, antarmuka, struktur data, dan algoritma yang akan digunakan. Pertimbangkan faktor seperti skalabilitas, modularitas, dan kegunaan kembali.
3. *Implementation*: Menulis kode untuk komponen perpustakaan berdasarkan spesifikasi desain. Mengikuti standar pengkodean, praktik terbaik, dan menjaga dokumentasi yang tepat.
4. *Testing*: Melakukan pengujian menyeluruh terhadap komponen perpustakaan untuk memastikan kebenaran, keandalan, dan kekokohan. Ini termasuk pengujian unit, pengujian integrasi, pengujian kinerja, dan keamanan.
5. *Deployment*: Mengemas komponen perpustakaan untuk didistribusikan dan diimplementasikan. Menyediakan dokumentasi, panduan, dan contoh yang sesuai agar pengguna dapat menggunakan perpustakaan dengan efektif.
6. *Maintenance*: Memantau penggunaan komponen perpustakaan, mengumpulkan umpan balik, dan menangani masalah atau bug yang dilaporkan. Merilis pembaruan atau versi baru sesuai kebutuhan untuk meningkatkan fungsionalitas, kinerja, dan kompatibilitas.

* 1. **Analisis Masalah**

Dalam proses analisis masalah dalam pengembangan komponen pustaka ini, penulis melakukan analisis terhadap komponen pustaka serupa yang ada. Penelitian ini bertujuan untuk mengeksplorasi pengalaman dan solusi yang telah diterapkan dalam konteks yang serupa. Dengan mempelajari komponen pustaka serupa, peneliti dapat mengidentifikasi masalah yang umum terjadi, hambatan yang dihadapi, serta solusi yang telah sukses diterapkan. Informasi ini sangat berharga dalam memahami panorama masalah dan mencari pendekatan yang efektif dalam pengembangan aplikasi yang sedang dirancang.

Selain itu, wawancara dengan *developer* yang relevan juga menjadi sumber informasi penting dalam analisis masalah. Melalui wawancara, penulis berkomunikasi langsung dengan Senior Developer, Junior Developer, dan Intern Developer yang terlibat dalam konteks yang relevan. Wawancara ini memberikan pemahaman yang lebih dalam tentang masalah yang dihadapi, kebutuhan yang harus dipenuhi, serta perspektif individu terkait dengan pengembangan aplikasi. Data yang diperoleh dari wawancara membantu dalam merumuskan pemahaman yang holistik tentang masalah yang ada.

* + 1. **Analisis Komponen Pustaka Serupa**
    2. **Analisis Berdasarkan Wawancara**
  1. **Perancangan**

# BAB 4

**HASIL DAN PEMBAHASAN**

# BAB 5

# REFERENCES