TUGAS PENDAHULUAN

PEMROGRAMAN PERANGKAT BERGERAK

MODUL XIV

DATA STORAGE BAGIAN 3 (API)

Disusun Oleh :

Maria Nathasya Desfera Pangestu / 2211104008

SE0601

Asisten Praktikum :

Muhammad Faza Zulian Gesit Al Barru

Aisyah Hasna Aulia

Dosen Pengampu :

Yudha Islami Sulistya, S.Kom., M.Cs.

PROGRAM STUDI S1 SOFTWARE ENGINEERING

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

TUGAS PENDAHULUAN

SOAL

1. Sebutkan dan jelaskan dua jenis utama **Web Service** yang sering digunakan dalam pengembangan aplikasi.
2. Apa yang dimaksud dengan **Data Storage API**, dan bagaimana API ini mempermudah pengelolaan data dalam aplikasi?
3. Jelaskan bagaimana proses kerja komunikasi antara klien dan server dalam sebuah Web Service, mulai dari permintaan (*request*) hingga tanggapan (*response*).
4. Mengapa keamanan penting dalam penggunaan **Web Service**, dan metode apa saja yang dapat diterapkan untuk memastikan data tetap aman?

Jawab

1. Dua jenis utama Web Service yang sering digunakan dalam pengembangan aplikasi yaitu:
   - RESTful Web Services (Representational State Transfer)
     Merupakan arsitektur perangkat lunak yang menjalankan sumber daya dengan metode HTTP seperti GET, POST, PUT, dan DELETE. Menggunakan format data seperti JSON atau XML dan lebih ringan dibandingkan SOAP. Serta lebih fleksibel dan mudah digunakan sehingga web services ini banyak digunakan dalam pengembangan aplikasi web modern karena lebih mudah diintegrasikan dengan aplikasi berbasis web dan mobile.
   - SOAP Web Services (Simple Object Access Protocol)
     Merupakan protokol yang lebih kompleks berbasis XML digunakan untuk pertukaran data antara sistem. Protokol ini menggunakan XML untuk mendefinisikan pesan dan biasanya beroperasi di atas protokol HTTP atau SMTP. SOAP memiliki standar yang ketat dan mendukung berbagai fitur seperti keamanan dan transaksi, sehingga sering digunakan dalam aplikasi bisnis yang membutuhkan integrasi yang kompleks dan aman.

2. Data Storage API yaitu jenis antarmuka pemrograman aplikasi (API) yang memungkinkan pengembang untuk menyimpan, mengambil, memperbarui, dan menghapus data dari penyimpanan yang dikelola aplikasi. Data dapat disimpan di berbagai lokasi, seperti basis data, file lokal, penyimpanan cloud, atau memori sementara. Secara sederhana, API menghubungkan aplikasi yang kita buat ke lokasi penyimpanan data, seperti database, penyimpanan cloud, atau sistem fitur. API mempermudah pengelolaan data dalam aplikasi dengan cara:
   - Abstraksi Proses Penyimpanan: API ini menawarkan fungsi yang mudah digunakan untuk membuat proses penyimpanan data lebih mudah. Pengembang dapat fokus pada logika aplikasi karena mereka tidak perlu khawatir tentang detail teknis sistem penyimpanan dasar.
   - Konsistensi dan Keamanan: Pengembang dapat menggunakan Data Storage API untuk memastikan bahwa data dikelola dengan konsisten dan aman. API menawarkan fitur keamanan yang mencegah akses yang tidak diinginkan ke data, dan mereka juga memungkinkan pengembang untuk menggunakan fitur otentikasi

dan otorisasi, seperti kunci akses atau token API, yang memastikan bahwa hanya pengguna atau aplikasi yang sah yang dapat mengakses data.

- Efisiensi: API sering dioptimalkan untuk melakukan tugas data seperti membaca, menulis, dan menghapus. Ini membuat aplikasi berjalan lebih cepat dan efisien. API Data Storage menyediakan fungsi standar untuk operasi CRUD, yang berarti membuat, membaca, mengubah, dan menghapus. Misalnya, membuat menambah data baru ke penyimpanan, membaca mengambil data yang ada, mengubah data yang ada, dan menghapus data dari penyimpanan. API menangani berbagai detail teknis, seperti sintaks database, manajemen koneksi, dan query.
- Interoperabilitas: API Penyimpanan Data memungkinkan aplikasi berinteraksi dengan berbagai jenis penyimpanan, baik lokal maupun cloud. Ini memberikan pengembang fleksibilitas untuk memilih solusi penyimpanan yang paling sesuai dengan kebutuhan aplikasi mereka seperti penyimpanan (SQL, NoSQL, penyimpanan cloud, atau file system) tanpa mengubah kode yang signifikan.
- Pengelolaan Data yang fleksibel: API ini sering menawarkan metode untuk melakukan operasi batch, yang memungkinkan pengembang melakukan berbagai tugas penyimpanan sekaligus. Ini meningkatkan efisiensi dan fleksibilitas dalam pengelolaan data.

3. Proses kerja komunikasi antara klien dan server dalam sebuah Web Service, mulai dari permintaan (*request*) hingga tanggapan (*response*) yaitu sebagai berikut:
    1) Klien Mengirim Permintaan (Request)
        a. Mekanisme: Klien, seperti aplikasi web, aplikasi mobile, atau alat lain, mengirimkan permintaan ke server menggunakan protokol HTTP/HTTPS.
        b. Komponen Utama Request:
            - URL: Alamat endpoint Web Service, misalnya https://api.example.com/users.
            - Metode HTTP: Jenis operasi yang diinginkan (misalnya, GET untuk membaca data, POST untuk menambahkan data, PUT untuk memperbarui, DELETE untuk menghapus).
            - Headers: Informasi tambahan seperti format data yang diinginkan (JSON/XML) atau token otentikasi.
            - Body (opsional): Data tambahan yang dikirimkan untuk operasi tertentu (misalnya, detail pengguna baru untuk operasi POST).
    2) Server Menerima dan Memproses Permintaan
        a. Penerimaan: Server menerima permintaan dan memverifikasi informasi dalam header atau body (misalnya, memvalidasi otentikasi).
        b. Rute atau Endpoint: Server mengarahkan permintaan ke layanan atau logika spesifik berdasarkan URL dan metode HTTP. Contoh: Permintaan GET /users diarahkan ke fungsi yang menangani pengambilan data pengguna.
        c. Pemrosesan Data: Server melakukan operasi yang diminta, seperti membaca data dari database, menjalankan logika bisnis, atau memproses informasi lainnya. Jika operasi melibatkan sumber daya eksternal, seperti basis data atau API lain, server melakukan komunikasi tambahan.
    3) Server Mengirim Tanggapan (Response)

a. Format Tanggapan: Server membungkus hasil operasinya dalam format yang telah disepakati (JSON, XML, atau lainnya).
b. Komponen Utama Response:
   - Kode Status HTTP (menunjukkan hasil operasi):
   - 200 (OK): Operasi berhasil.
   - 201 (Created): Data berhasil dibuat.
   - 400 (Bad Request): Permintaan salah atau tidak valid.
   - 401 (Unauthorized): Otentikasi gagal.
   - 404 (Not Found): Sumber daya tidak ditemukan.
   - 500 (Internal Server Error): Kesalahan pada server.
   - Headers: Informasi tambahan, seperti jenis data yang dikirimkan.
   - Body: Data hasil operasi, jika ada (misalnya, data pengguna dalam permintaan GET /users).

4) Klien Menerima dan Mengolah Tanggapan
   a. Penerimaan Data: Klien menerima tanggapan dari server.
   b. Pemrosesan: Klien membaca body tanggapan, memproses data yang diterima, atau menampilkan pesan kesalahan kepada pengguna jika terjadi kegagalan.
   c. Tindakan Lanjutan: Berdasarkan hasil tanggapan, klien dapat mengambil langkah berikutnya, seperti memperbarui UI atau mengirim permintaan tambahan.

4. Karena Web Service sering digunakan untuk mengirimkan data sensitif antara klien dan server maka keamanan dalam penggunaan Web Service sangat penting. Jika tidak aman maka bisa disadap, diubah, atau disalahgunakan oleh pihak yang tidak berwenang. Serangan man-in-the-middle (MITM), injeksi SQL, atau penyamaran identitas (spoofing) adalah contoh ancaman keamanan yang dapat mengancam integritas, kerahasiaan, dan ketersediaan data. Keamanan dalam Web Service penting karena:
   - Melindungi Data Sensitif pada Web Service: Selama pengiriman dan penyimpanan, data sensitif seperti kata sandi, informasi kartu kredit, dan data pribadi pengguna harus aman.
   - Mencegah Akses Tidak Sah: Layanan dan data hanya dapat diakses oleh pengguna atau sistem yang berwenang.
   - Menjamin Integritas Data: Untuk memastikan bahwa data tidak diubah selama pengiriman.
   - Kesesuaian dengan Peraturan: Banyak peraturan, seperti GDPR, HIPAA, atau PCI-DSS, menuntut standar keamanan tinggi dalam penanganan data.
   - Kepercayaan Pengguna: Keamanan yang baik meningkatkan kepercayaan pengguna terhadap aplikasi atau layanan.

   Metode yang dapat diterapkan untuk memastikan data tetap aman yaitu seperti:

   1) Enkripsi Data
      - Lapisan Transportasi Data (TLS): menggunakan HTTPS untuk mengamankan komunikasi antara klien dan server dan mengenkripsi data selama transmisi.

- Enkripsi End-to-End: Data tetap terenkripsi dari klien hingga server, sehingga pihak ketiga tidak dapat membacanya.

2) Otentikasi (Otentikasi)
   - API Key: Memberikan token unik kepada klien untuk mengidentifikasi permintaan yang sah.
   - OAuth: Protokol otorisasi standar yang memungkinkan akses aman ke API pihak ketiga tanpa membagikan kredensial.
   - JSON Web Tokens (JWT): Digunakan untuk menjamin bahwa pengguna dapat menggunakan token dengan masa berlaku tertentu.
3) Otorisasi (Authorization)
   - Role-Based Access Control (RBAC): Memberikan hak akses berdasarkan peran pengguna.
   - Access Control Policies: mengatur data atau fitur yang dapat diakses oleh klien tertentu.Metode Keamanan Data Web Service
4) Validasi Input
   - Melakukan sanitasi dan validasi input untuk mencegah serangan injeksi seperti SQL Injection atau XSS.
5) Penggunaan API firewall
   - API firewall melindungi endpoint dari serangan brute force atau DDoS.
6) Batas Rate dan Throttling
   - Digunakan untuk mencegah penyalahgunaan dengan membatasi jumlah permintaan klien dalam jangka waktu tertentu.
7) Token Expiration dan Revocation
   - Token otentikasi harus memiliki masa berlaku untuk menghindari penyalahgunaan setelah token dicuri.
8) Audit dan Logging
   - Mencatat semua permintaan dan respons untuk mengidentifikasi tindakan mencurigakan atau serangan keamanan.
9) Keamanan Data Server
   - Enkripsi data yang disimpan (enkripsi di tengah jalan) dan memastikan bahwa pembaruan perangkat lunak server terus dilakukan untuk mencegah kerentanan.
10) Melaksanakan CORS(Cross-Origin Resource Sharing)
   - Mengatur domain mana yang dapat mengakses Web Service untuk melindungi dari serangan dari sumber yang tidak dipercaya.