TUGAS PENDAHULUAN
PEMROGRAMAN PERANGKAT BERGERAK

MODUL XIV

DATA STORAGE API

Disusun Oleh :

Devrin Anggun Saputri / 2211104001

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

a. **Sebutkan dan jelaskan dua jenis utama Web Service yang sering digunakan dalam pengembangan aplikasi.**

1. **SOAP (Simple Object Access Protocol)**

   SOAP adalah protokol berbasis XML untuk pertukaran data dalam lingkungan terdistribusi, menggunakan standar yang lebih ketat dibandingkan dengan jenis web service lainnya. **SOAP** cocok untuk aplikasi dengan **proses bisnis kompleks** dan **keamanan tinggi**, meskipun lebih berat.

   Ciri-Ciri SOAP:

   - **Berbasis XML:** Semua data dikirim dalam format XML yang terstruktur.
   - **Protokol Standar:** Menggunakan protokol seperti HTTP, SMTP, dan FTP untuk komunikasi.
   - **Memiliki WSDL (Web Services Description Language):** WSDL mendeskripsikan layanan web (metode, parameter, dan data) sehingga klien tahu bagaimana berinteraksi dengan layanan tersebut.
   - **Keamanan yang Baik:** Mendukung fitur keamanan seperti WS-Security untuk enkripsi dan otentikasi.

2. **REST (Representational State Transfer)**

   REST adalah arsitektur berbasis **HTTP** yang lebih sederhana dan fleksibel dibandingkan SOAP, sering digunakan dalam pengembangan aplikasi modern. **REST** lebih sering digunakan karena **sederhana**, **cepat**, dan fleksibel, terutama dalam pengembangan aplikasi modern berbasis web dan mobile.

   Ciri-Ciri REST:

   - **Berbasis Resource:** Setiap resource memiliki URL unik.
   - **Format Data Beragam:** Menggunakan JSON, XML, HTML, atau teks biasa, dengan JSON menjadi pilihan utama karena lebih ringan.
   - **Stateless:** Tidak menyimpan data sesi di server, setiap request memiliki semua informasi yang diperlukan.
   - **Operasi CRUD:** Biasanya menggunakan metode HTTP seperti:
     a. **GET:** Membaca data
     b. **POST:** Menambahkan data
     c. **PUT:** Memperbarui data
     d. **DELETE:** Menghapus data

b. Apa yang dimaksud dengan Data Storage API, dan bagaimana API ini mempermudah pengelolaan data dalam aplikasi?

Data Storage API adalah antarmuka pemrograman aplikasi (Application Programming Interface) yang memungkinkan pengembang untuk menyimpan, mengelola, dan mengambil data secara efisien dalam suatu aplikasi. API ini dapat beroperasi di berbagai jenis penyimpanan, seperti:

1. **Local Storage**: Penyimpanan lokal di perangkat pengguna, seperti *localStorage* di web browser atau penyimpanan file lokal di aplikasi mobile.
2. **Cloud Storage**: Penyimpanan data di server jarak jauh melalui layanan berbasis cloud seperti **Google Cloud Storage**, **Amazon S3**, atau **Firebase Storage**.
3. **Database API**: API yang memungkinkan akses ke database untuk mengelola data dalam format terstruktur, seperti SQL atau NoSQL.

Bagaimana API mempermudah pengelolaan data aplikasi:

1. **Abstraksi Penyimpanan Data**
   - Data Storage API menyembunyikan kompleksitas teknis dalam mengakses data di backend atau di penyimpanan lokal.
   - Pengembang cukup memanggil fungsi API tertentu seperti `saveData()`, `retrieveData()`, atau `deleteData()` tanpa perlu mengetahui detail implementasi penyimpanan data.
2. **Kemudahan Integrasi**

   Data Storage API menyediakan antarmuka standar yang mudah diintegrasikan ke dalam aplikasi. Misalnya:
   - API berbasis cloud (contoh: Firebase Storage) memungkinkan aplikasi mobile dan web untuk menyimpan file atau data secara langsung ke cloud.
   - Local Storage API di web browser menyimpan data sederhana seperti preferensi pengguna.
3. **Penyimpanan Fleksibel**
   - Data Terstruktur: Menggunakan database relasional seperti MySQL atau NoSQL seperti MongoDB.
   - Data Tidak Terstruktur: File seperti gambar, video, atau dokumen.
   - Data Ringan: LocalStorage API untuk menyimpan data berbasis key-value seperti token pengguna.
4. **Akses Cepat dan Skalabilitas**
   - API ini memungkinkan pengambilan dan penyimpanan data yang cepat, baik dalam penyimpanan lokal maupun cloud.
   - Cloud Storage API dapat meningkatkan skalabilitas dengan menyesuaikan kapasitas penyimpanan sesuai kebutuhan
5. **Manajemen Keamanan Data**
   - Data Storage API sering kali dilengkapi dengan fitur keamanan seperti otentikasi (contoh: OAuth), enkripsi data, dan kontrol akses (ACL).
   - Ini membantu menjaga kerahasiaan dan integritas data dalam aplikasi.

Contoh Penggunaan:

- **Local Storage API di Web**

```
// Simpan data
localStorage.setItem("username", "JohnDoe");

// Ambil data
const user = localStorage.getItem("username");
console.log(user); // Output: JohnDoe
```

- **Cloud Storage API (Firebase Storage)**

```
import 'package:firebase_storage/firebase_storage.dart';

// Upload file
Future<void> uploadFile(String filePath) async {
  File file = File(filePath);
  try {
    await FirebaseStorage.instance
        .ref('uploads/image.jpg')
        .putFile(file);
  } catch (e) {
    print('Error: $e');
  }
}
```

- **Database API**

```
// Menggunakan MongoDB API untuk menambahkan data
const MongoClient = require('mongodb').MongoClient;

const uri = "your_mongodb_uri";
const client = new MongoClient(uri);
async function run() {
  try {
    await client.connect();
    const database = client.db("exampleDB");
    const users = database.collection("users");

    const newUser = { name: "John Doe", age: 30 };
    await users.insertOne(newUser);
  } finally {
    await client.close();
  }
}
run();
```

c. Jelaskan bagaimana proses kerja komunikasi antara klien dan server dalam sebuah Web Service, mulai dari permintaan (request) hingga tanggapan (response).

Proses komunikasi antara **klien** dan **server** dalam sebuah **Web Service** melibatkan pertukaran data yang menggunakan protokol komunikasi standar seperti **HTTP**. Proses ini terdiri dari **permintaan (request)** dari klien dan **tanggapan (response)** dari server. Berikut adalah penjelasan langkah demi langkah proses kerjanya:

1. **Klien**: Mengirim request ke server.
   → **Server**: Menerima request.
2. **Server**: Memproses request (validasi, logika bisnis, database).
3. **Server**: Mengirim response kembali ke klien.
4. **Klien**: Menerima response dan memprosesnya.

d. Mengapa keamanan penting dalam penggunaan Web Service, dan metode apa saja yang dapat diterapkan untuk memastikan data tetap aman?

Keamanan sangat penting dalam penggunaan **Web Service** karena data yang ditransmisikan antara **klien** dan **server** sering kali bersifat sensitif, seperti informasi pengguna, kredensial login, atau data transaksi. Jika keamanan tidak dijaga, dapat terjadi berbagai risiko, seperti:

1. **Kebocoran Data**: Data sensitif dapat diakses atau dicuri oleh pihak yang tidak berwenang.
2. **Man-in-the-Middle Attack (MITM)**: Penyerang dapat memotong komunikasi antara klien dan server dan mencuri atau memodifikasi data.
3. **Penyalahgunaan Akses**: Jika autentikasi lemah, pihak yang tidak berwenang dapat mengakses Web Service.
4. **Injeksi dan Serangan Berbasis Input**: Serangan seperti **SQL Injection** atau **Cross-Site Scripting (XSS)** dapat membahayakan data dan sistem.
5. **Integritas Data**: Data yang dikirim bisa diubah oleh penyerang jika tidak ada proteksi.

Metode Keamanan untuk Menjaga Keamanan Web Service:

1. **Menggunakan HTTPS (SSL/TLS)**
   - Menggunakan protokol **HTTPS** (Hypertext Transfer Protocol Secure) mengenkripsi data yang dikirim antara klien dan server.
   - SSL/TLS mencegah serangan **Man-in-the-Middle** dan memastikan data tidak mudah disadap.
   - **Contoh**: URL Web Service harus menggunakan `https://`.
2. **Autentikasi Pengguna**
   - API Key: Token unik untuk setiap pengguna yang dikirim bersama request.
   - Basic Authentication: Menggunakan kombinasi username dan password (biasanya dikodekan dalam base64).
   - OAuth 2.0: Protokol autentikasi yang aman dan populer untuk memberikan akses berbasis token.
   - JWT (JSON Web Token): Token berbasis JSON yang digunakan untuk autentikasi dan menjaga integritas data.
3. **Otorisasi (Authorization)**
   - Setelah autentikasi, otorisasi digunakan untuk memastikan pengguna hanya memiliki akses sesuai dengan haknya.
   - Role-Based Access Control (RBAC) atau Access Control List (ACL) sering digunakan untuk membatasi akses ke resource tertentu.

4. **Enkripsi Data**
    - Enkripsi digunakan untuk melindungi data sensitif, baik saat dikirim (in-transit) maupun saat disimpan (at-rest).
    - In-transit encryption: Menggunakan TLS/SSL saat mengirim data melalui jaringan.
    - At-rest encryption: Mengenkripsi data di database atau penyimpanan cloud.
5. **CORS (Cross-Origin Resource Sharing) Policy**
    - CORS digunakan untuk membatasi dari mana permintaan ke Web Service bisa berasal.
    - Hanya klien dari domain yang terpercaya yang diizinkan untuk mengakses API

      Contoh:

      ```
      Access-Control-Allow-Origin: https://trusted-domain.com
      ```