NSS – SOA a REST

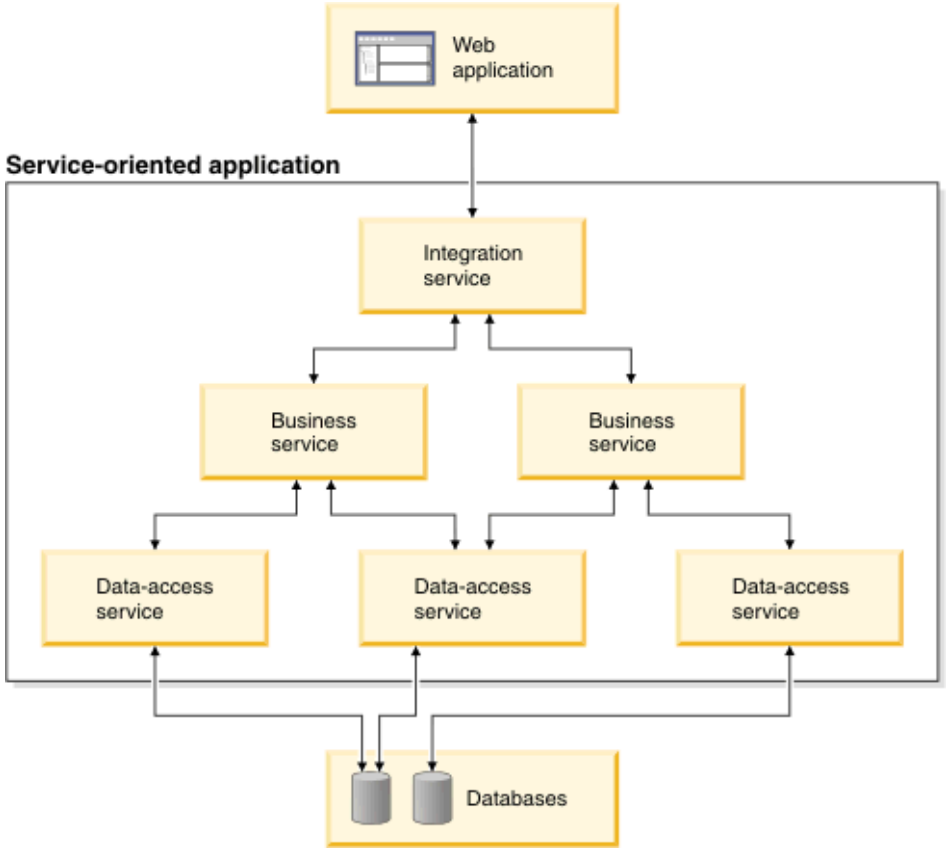
11. LECTURE

MARTIN TOMASEK

SOA (Servisně Orientované Architektury)

1. Sada principů a metodologií
2. Skládá se z nezávislých komponent, které poskytují specifické služby.
3. Služby komunikují s okolními službami.
4. Obvykle spolu komunikují pomocí HTTP protokolu.
5. Změna v jedné službě nemusí ovlivnit klienty či ostatní služby.
6. Služby na sobě nejsou navzájem technologicky závislé.
7. Služby vystavují rozhraní, které slouží ke komunikaci s nimi. Například API (Application Programming Interface)

SOA Ukázka



Převzato z: https://www.ibm.com/support/knowledgecenter/en/SSMQ79_9.5.1/com.ibm.egl.pg.doc/topics/pegl_serv_overview.html

HTTP

1. Hypertext Transfer Protocol (1.1 a 2)
2. Aplikační protokol pro přenos hypertextu – od roku 1990. Nad TCP/IP protokolem
3. Jedná se request/response protocol
4. Zdroj se identifikuje pomocí URI a Metody (GET, PUT, POST, DELETE,..).
5. Response code reprezentuje stav, který způsobil request na serveru (https://www.w3.org/Protocols/rfc2616/rfc2616-sec6.html#sec6.1.1)
6. V rámci HTTP lze přenášet obrázky, data (XML, JSON), XHTML, webové stránky, zdrojové kódy, soubory a pod
7. Zabezpečená varianta je HTTPS

REST

1. Representational State Transfer
2. Jedná se o architektonický styl, který je založen na principech, podle kterých jsou definovány a adresovány zdroje.
3. Pokud aplikace splňuje všechny principy, pak ji označujeme jako RESTfull
4. Funkcionality jsou rozděleny do oddělených zdrojů
5. Každý zdroj je unikátně adresovaný a určitelný pomocí jeho metody
6. Protokol je client/server a je bezestavový, vrstevnatý a podporuje cachování.

RESTfull

1. Jednotné rozhraní
2. Client-server
3. Bezestavový
4. Cachovaný
5. Vrstevnatý
6. Executovaný kód na vyžádání

https://restfulapi.net/rest-architectural-constraints/#uniform-interface

JSON a XML

Přenos dat je realizován ve formátu JSON nebo XML

Konverze mezi těmito formáty je podporována, nicméně může být ztrátová.

Konkrétní formát si vynucuje server, popřípadě určuje klient pomocí správných HTTP hlaviček

JSON (JavaScript Object Notation) – Popsán JSON schema. Lehčí než XML, méně objemnější.

XML – eXtensive Markup Language – popsán XML schema či DTD. Umožňuje specifikovat více atributů a vlastností. Obvykle se používá i pro SOAP služby.

JSON - Example


```
{
    "pageNumber": 0,
    "pageCount": 1,
    "pageSize": 3800,
    "totalItemCount": 3800,
    "items": [
        {
            "id": 40000000001098,
            "location": {
                "lat": 47.1879175,
                "lng": 20.0047114
            },
            "type": "ATM",
            "name": "Városi Sportcsarnok",
            "address": "Kossuth Lajos tér 18.",
            "city": "Abony",
            "postCode": "2740",
            "country": "HU",
            "bankCode": "---"
        },
```

XML - Example


```
<o>
  <items>
    <e>
      <address>Kossuth Lajos tér 18.</address>
      <bankCode>---</bankCode>
      <city>Abony</city>
      <country>HU</country>
      <id>40000000001098</id>
      <location>
        <lat>47.187916</lat>
        <lng>20.004711</lng>
      </location>
      <name>Városi Sportcsarnok</name>
      <postCode>2740</postCode>
      <type>ATM</type>
    </e>
    <e>
      <accessType>nepřetržitě</accessType>
      <address>Družstevní 421/1</address>
      <atmNumber>6321</atmNumber>
      <bankCode>0800</bankCode>
      <city>Adamov</city>
      <country>CZ</country>
      <id>6321</id>
      <location>
        <lat>49.291096</lat>
        <lng>16.665266</lng>
      </location>
      <name>Budova města</name>
      <postCode>67904</postCode>
      <region>Jihomoravský kraj</region>
      <type>ATM</type>
    </e>
```

API Dokumentace

Služby je potřeba správně dokumentovat.

Je nutné poskytnout vývojáři zpětnou vazbu a možnost vyzkoušet si volání.

Je nutné udržovat informace o API pro jeho následující rozvoj.

K tomuto účelu lze využít APIARI:

- Lze propojit s repozitářem
- Umožňuje generování dokumentace na základě MarkUP language.
- Slouží pro návrh a jeho následnou udržitelnost.

Více na: https://apiary.io/

API Gateway

1. Slouží jako vstupní brána pro REST služby.
2. Zajišťuje základní orchestraci, ověření přístupu, logování, API Managementu, omezení počtu volání,..
3. Zajišťuje směrování na poptávané služby, překlad hlaviček, úpravy v těle zpráv apod.
4. Zajišťuje základní bezpečnostní kontroly
    1. Vyhodnocení rizikovosti zpráv, jejich inspekci a validaci. (Velikost zpráv, SQL Injection, Antivirová kontrola, validace vůči schema, test na JSON/XML bomby)
    2. Vyhodnocení oprávněnosti volání (Token
    3. Vyhodnocení aplikace, která volání provedla (Two-way-SSL)
5. Může jít jak o komerční produkty, tak produkty vlastní výroby
    1. Axway API Gateway (https://www.youtube.com/watch?v=Dq7GwLvRhLg / https://www.axway.com/en/enterprise-solutions/api-management-plus#tablist1-tab1)
    2. WebAPI – (Confidential) – https://developers.csas.cz / https://developers.erstegroup.com/
6. Špatné použití může vést k antipaternům.
7. Nepoužití vede k vyšším nárokům na aplikace, které služby poskytují. Tyto aplikace pak implementují tyto nároky obvykle nekoncepčně.

Oauth Token a JWT

1. Slouží k identifikaci uživatele a jeho vlastností v konkrétním systému (Scope, expiration_time,…)
2. Tokeny jsou vydané vždy pro určitou aplikaci a musí být používány pouze v této aplikaci.
3. Tokeny jsou změť znaků, které je nutné přeložit / dešifrovat. (Pozor na výkon).
4. Oauth Token
   1. Předává se na FE, avšak ten není schopný z tohoto tokenu nic poznat.
   2. K získání informací z tohoto tokenu je nutné použít bezpečnostní komponentu dané organizace, která token vydala.
   3. Při každém volání API je nutné token ověřit.
5. JWT token
   1. Nese informaci o uživateli + mnoho dalších věcí.
   2. Lze také předat na FE.
   3. Není nutné oslovovat komponentu, která ho vydala.
   4. Platnost tokenu se řídí platností v něm samém.

Sandbox a Mocky

1. Sandbox – pískoviště kde lze vyzkoušet integraci na API bez ovlivnění systémů.
    1. Řeší problém se stabilitou neprodukčních prostředí.
    2. Řeší problém s integrací na neprodukční i produkční prostředí v prvotní fázi vývoje.
2. Mocky – statické či semi-statické zprávy, které API vrací.
    1. Řeší základní problém s datovou kvalitou.
    2. Řeší integrační problém na neprodukčních prostředí.
    3. Lze je využít k prodeji výrobku, aniž by se do něj uživatel musel přihlásit.
    4. Mockovat lze jak produkční tak neprodukční prostředí.
    5. Čím kvalitnější mock -> tím lepší vývoj -> tím větší a mnohdy nekoncepční nároky na mocky
3. Nezbytný nástroj pro vývoj 3. stran.
4. Slouží k urychlení integračního vývoje.