


# Základy mimo banku


## 1. HTTP protokol

- a. co je to (jak vypadá - payload, http headers, http response code, content type)
- b. co je webservice (microservice)
- c. autorizace, autentizace
  - i. co to je
  - ii. jaké mechanismy se mohou používat
  - iii. jaký je rozdíl mezi autorizací a autentizací
- d. HTTPS (SSL, TLS, mTLS) 
  - i. obecné povědomí o certifikátech (serverový / klientský certifikát)
  - ii. privátní / veřejný klíč (PKI - public key infrastructure)
- e. Metody (get, post, put, patch, delete) dle RFC
  - i. jak se komunikuje pomocí SOAP
  - ii. jak se komunikuje u REST služeb (jaké se používají metody a jaké mají definice / omezení)
  - iii. příklady, jak mohou vypadat REQ pro jednotlivé metody
- f. HTTP/2
- g. problematika CORS (cross origin resource sharing)
- h. seznámení se s CLI nástrojem cURL / SOAP UI Free
- i. úkoly viz níže (pomocí nástroje cURL vytvořit REQ pro každou metodu - lze se inspirovat na <https://petstore.swagger.io>, atd.) <https://httpbin.org/>

## 2. práce s XML a jednoduchá XSL transformace

- a. co je xml a jak vypadá
- b. co je to XPATH
- c. co je to XSLT
- d. úkoly viz níže (vytvořit jednoduchou DB v XML, zflakovat atd.)

## 3. SOAP

- a. jak vypadá SOAP 1.1 a 1.2 (rozdíly) - Banka používá SOAP 1.1 (definice SOAP envelope)
- b. definice SOAP rozhraní (WSDL, XSD)
- c. WS security
- d. co jsou to XSD, k čemu se používají, jak vypadají namespaces
- e. vytvoření jednoduchého XSD, WSDL s více operacemi
- f. práce s XPATH (již z okruhu 1) 
  - i. jednoduchý pick
  - ii. komplikovanější řetězec
- g. práce se SOAP UI free
  - i. vytvoření projektu + nastavení (kde nastavit WSS)
  - ii. provolání služeb
  - iii. test case
  - iv. mock server
  - v. jednoduché scripty
- h. úkoly:

- i. vytvořte 2 XSD - vycházejte z prvního úkolu
  - 1. první XSD definuje vaši DB
  - 2. druhé XSD definuje výsledné (zflatované) XML
- ii. zkuste vytvořit WSDL (Soap v1.1) ComputerComponents\_v1, které obsahuje 2 operace GetAllComponents a GetFlatStructure s odkazy na dříve vytvořené XSD (definici pro REQ a RES)
- iii. WSDL si nainportujte do SOAP UI a zkuste si to otestovat

## 4. REST

- a. jak vypadá REST služba (jak se liší oproti SOAP), v jakém formátu chodí data (JSON, XML)
- b. jak správně používat metody
- c. popis REST rozhraní
  - i. JSON schema
  - ii. Swagger 2.0 / OpenAPI specification 3.0
  - iii. Vytvoření jednoduchého rozhraní (YAML)
- d. práce s POSTMANem / SoapUI Free
- e. vytvoření JSON a práce s JSON path
  - i. jednoduchý pick
  - ii. komplikovanější řetězec
- f. **úkoly:**
  - i. Vytvořte YAML (OAS3.0) rozhraní se stejnou strukturou jako XML z prvního úkolu (metoda GET)
  - ii. vytvořené rozhraní obohaťte o další 3 metody (vytvoření nové položky, úpravu položky a smazání položky). Navrhněte podobu REQ a následně i RES

## 5. OAUTH2

- a. základní mechanismy

## 6. Javascript (pro potřeby XML2JSON)

- a. základní syntaxe
- b. proměnné, podmínky, cykly
  - i. jednoduché datové typy
  - ii. pole (práce s polem)
  - iii. objekty (práce s objekty)

## 7. Práce s GITem a všeobecně VCS (používáme Bitbucket)

- a. co je to git
- b. commit a Kostičův oblíbený amend :)
- c. pull & push (remote repository)
- d. branches a mergování

## Práce v Bance

- 1. Jednotlivá prostředí a jejich rozdíly
  - a. GLOW
  - b. TDA
- 2. Co jsou to ITIM ROLE, jak používat různé user accounts na různých prostředích

3. základy s GW (Policy Manager)
  - a. jak se tvoří služba
    - i. SOAP (z WSDL)
    - ii. REST
  - b. co je to assertion
    - i. základní sada
    - ii. custom dodané pro Banku (swagger validation, XML2JSON)
  - c. co jsou to CWP
  - d. vytvoření jednoduché služby a její provolání (zadání máme na conflu)
    - i. REST
    - ii. SOAP
  - e. jak se používá debugger
  - f. základy transformací (XSLT, XML2JSON)
    - i. úpravy a převod requestů
      1. REST2SOAP
      2. SOAP2REST
      3. SOAP2SOAP
      4. REST2REST
4. všeobecný přehled infrastruktury banky
  - a. co je FE
  - b. co je BE
  - c. bezpečnost v bance
5. custom ČSOB framework na GW
  - a. inicializace služby
  - b. inicializace logování
  - c. logování samotné
  - d. error handling
6. Jak funguje logování
  - a. různé typy logů a rozdíly logování na různých instancích
7. Developer portal
  - a. co je to
  - b. k čemu se používá
  - c. jak vypadá API Key
  - d. jak integrovat portál do BankAPI služby
8. Práce s GMU
  - a. jak exportovat
  - b. jak importovat
9. Tvorba změnovek

# Úkoly

## Mimo banku - část 1.

1. Zaznamenejte pomocí XML nějakou malou databázi třeba s počítačovým komponentama/příslušenstvím atd – máte třeba desktop počítač, tak z čeho se skládá, nějak to hezky rozdělit na **komponenty** typu disk/paměti/grafika atd, na **periferie** typu klávesnice/myš atd., **příslušenství** jako židle, stůl apod. Nehleďte v tom moc vědu, jde o rozdělení na nějaké základní “balíky” toho co využíváte a následně to zaznamenejte do XML včetně nějakých detailů jako značka/kapacita/cokoli vás napadne. Jediná podmínka je tady, že součástí každé té součásti bude cena – tu si jakkoli vymyslete, nemusí být vůbec reálná, ale musí to být pouze a jen číslo.
  - pokud nemáte desktop, zamyslete se nad notebookem co obsahuje a udělejte to v podobném duchu jako s desktopem popisovaným výše – paměti, disk, display, klávesnice, grafika,...
  - inspirujte se, co se týká rozdělení, třeba nějakým větším českým e-shopem s počítačovým věcmi, lze to rozdělení od nich okopírovat
  - příklad vaší zjednodušené DB (udělejte si to po svém jak je libo) něco jako:

```
<?xml version="1.0" encoding="UTF-8"?>
<computer>
  <components>
    <component>
      <name>disk</name>
      <description>Seagate</description>
      <type>Ironwolf</type>
      <capacity>8000</capacity>
      <price>5025</price>
    </component>
    <component>
      ...
    </component>
    ...
  </components>
  <accessories>
    <acc>
      <name>židle</name>
      <description>Spinalis</description>
      <price>34000</price>
    </acc>
    ...
  </accessories>
</computer>
```

2. Projděte si v XML i znaky, které jsou nevalidní + jaké znaky je potřeba escapovat – např. jak zapsat &, “ a podobně.

3. Vaši databázi, kterou jste si vytvořili v bodu 2, udělejte pomocí XSLT “více flat” ve smyslu že vezmete vaší DB, aplikujete XSLT a výstupem bude něco jako

```
<flatComputer>
  <part>disk Seagate Ironwolf 8000 – 5025,-</part>
  ...
  <part>židle Spinalis – 34000,-</part>
  ...
```

</flatComputer>

Tzn. Vše bude rovnou na druhé úrovni, spojíte ty názvy/description/type/ “pomlčka” price (např. disk Seagate Ironwolf 8000 – 5025,-) a vytvoříte “placatější” databázi.

K tomu XSLT doporučuji využívat nějaké free nástroje co jsou online, já mám třeba rád docela cokoli co je zde

<https://www.freeformatter.com/xsl-transformer.html>, využívám tam i ty validátory atd. co tam můžete dohledat v levém sloupečku

4. Upravte vaše XSLT tak, aby se ve flat databázi zobrazovaly pouze položky, které stojí méně než 5000 (porovnávejte v xslt *price* oproti nastavenému thresholdu, zde těch 5000 – pokud máte ceny jen nad 5000 nebo naopak pod 5000, použijte práh takový, aby výstup byl jen podmnožinou původní množiny, ale zároveň počet položek  $\geq 2$ ).
5. Upravte XSLT pro původní flat databázi tak, aby se zobrazovaly pouze položky, jejichž název obsahuje písmeno „a“ (je jedno zda velké nebo malé).

## Mimo banku - část 2.

- 1.