

IŠ PRAEITOS PASKAITOS

- Daugiausiai klausimų dėl to, kad prieš tai nebandėte atlikti kokios nors užduoties
 - medžiaga yra koncentruota ir turi tam tikrą eilės tvarką, ir jei kažką praleidžiate, vis tiek teks sugrįžti
 - taigi jeigu nesuprantate užduoties - klauskite



IŠ PRAEITOS PASKAITOS

- React "key" padeda React'ui nuspresti, kas pasikeitė DOM medyje, kad galetų atnaujinti tai, ką reikia
 - naudoti, pvz., generuojant įvairius sąrašus
 - unikalus tame masyve (ne visame puslapyje)

```
class Vaisiai extends Component {  
    render() {  
        var vaisiai = this.props.vaisiai.map(vaisius => {  
            return (<div className="col-sm-4" key={vaisius.pavadinimas}>  
                <Vaisius paveikslukas={vaisius.paveikslukas} ..>  
            </div>);  
        })  
        ..>
```





AKADEMIJA.IT
INFOBALT IR TECH CITY

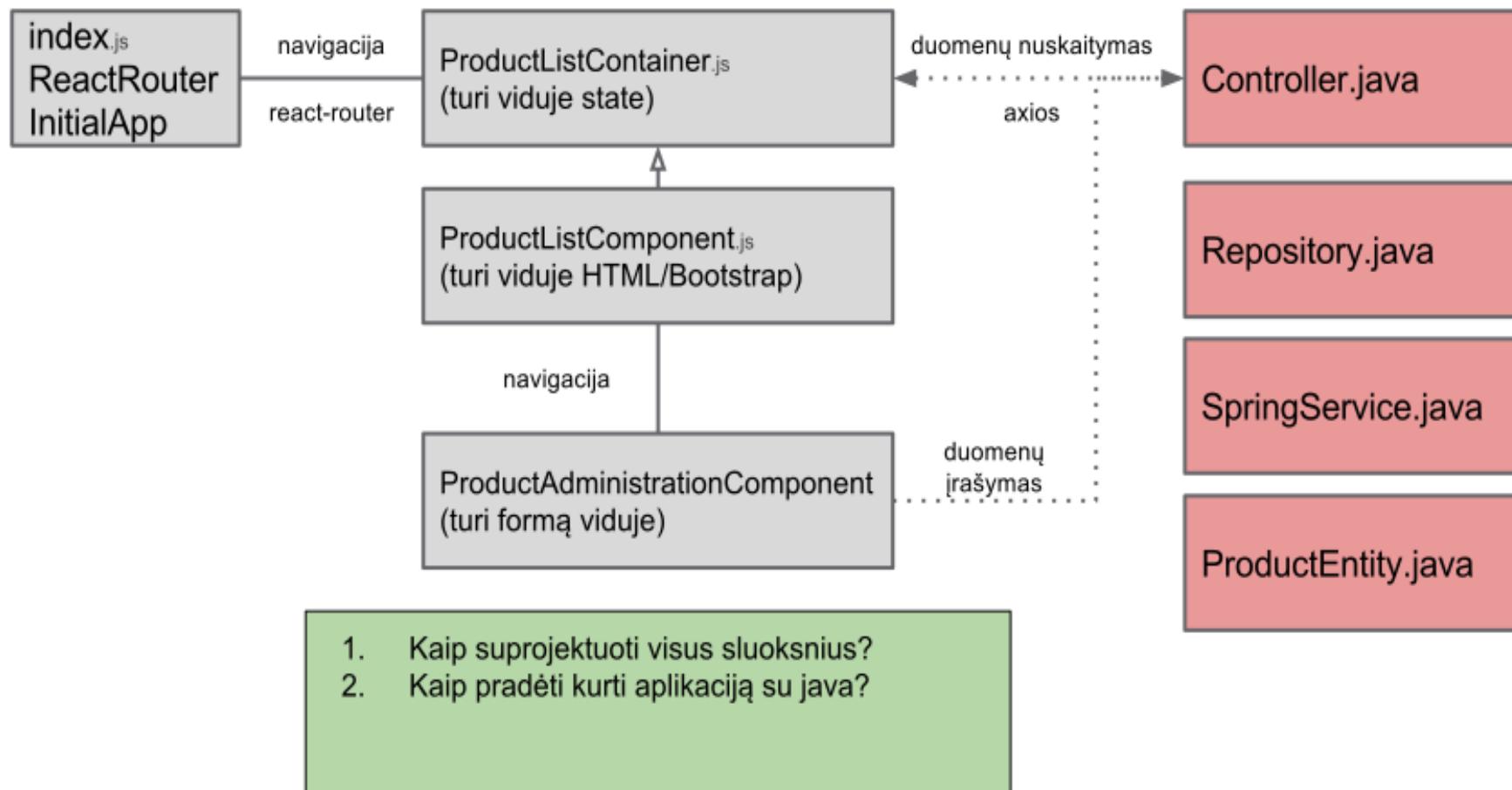
XML. KARKASAI. SISTEMŲ ARCHITEKTŪROS

Andrius Stašauskas

andrius@stasauskas.lt

<http://stasauskas.lt/itpro2018/>

KĄ JAU MOKAME IR KO DAR NE



PRIEŠ NAUDOJANT JAVA

- Ką reikia išmokti:
 - JSON
 - Rest API
 - XML/namespaces
 - kokią funkciją atlieka kokie karkasai
 - daugiasluoksnė architektūra
 - React pozicija joje
 - Tomcat
 - Maven - java "paketų manageris"
 - Spring



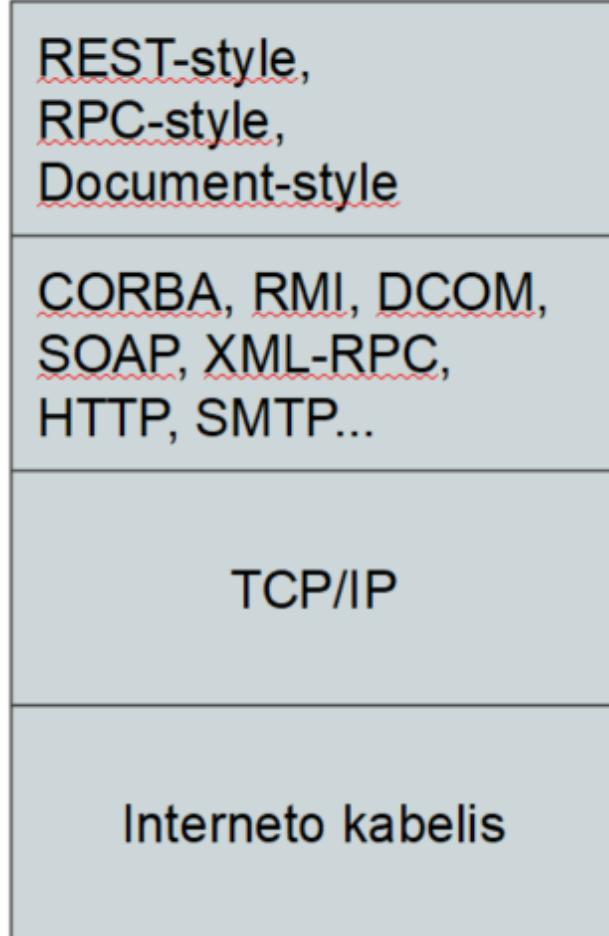
TURINYS

- Rest API
- Duomenų reprezentavimo formatai
- XML ir vardų erdvės (zonos/namespaces)
- Karkasai
- Sistemų architektūros



REST API

KOMUNIKACIJOS PROTOKOLAI



Pranešimų siuntimo stiliai

Komunikacijos ir serializacijos protokolai

Duomenų transportas



TERMINAI

- Pranešimai/Užklausos (angl. message/requests) - tinkle esančios programos bendrauja siūsdamos viena kitai pranešimus/užklausas.
- Galinis taškas (angl. endpoint) - tai URL adresą turintis servisas, kurį gali pasiekti klientas.
- URL (angl. Uniform Resource Locator) - tai nuoroda/adresas į kažkokį resursą tinkle, nurodant būdą, kaip tą resursą pasiekti.
- URI (angl. Uniform Resource Identifier) - tai nuoroda/adresas į kažkokį resursą tinkle.



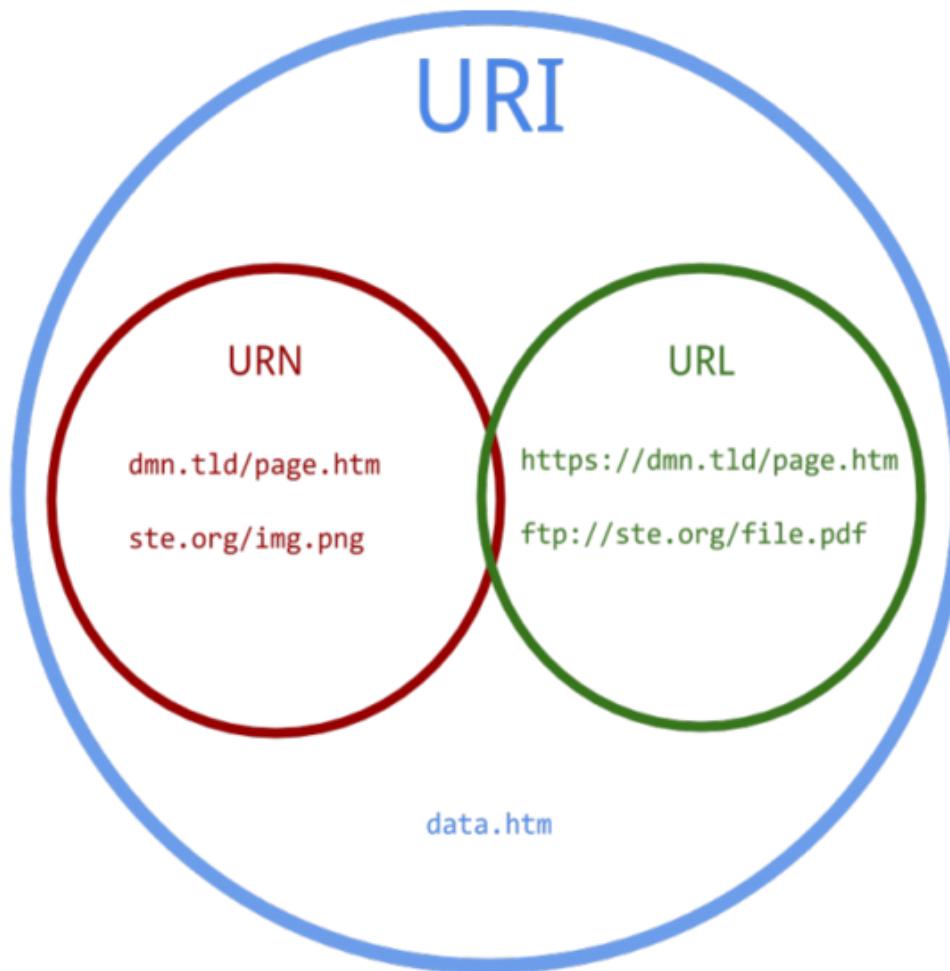
URL

- Bendrinė URL schema:

```
schema : [ //mašinosvardas [:portas] ] [/] kelias [ ?parametrai ] [#fragmentas]
```

- Schema - nurodo protokolą, kuriuo bus pasiekiamas resursas. Galimi protokolai: HTTP, FTP ir pan.
- Mašinos vardas - unikalus serverio vardas (registruotas vardas arba IP adresas)
- Portas - skaičius, nurodantis loginę jungtį (programos adresą) toje mašinoje.
- Kelias - relatyvus resurso adresas
- Parametrai - užklausos parametrai
- Fragmentas - nuoroda/adresas dokumento lygyje

URI, URL, URN



DUOMENŲ REPREZENTAVIMO FORMATAI

- Duomenų reprezentavimo formatai:
 - XML
 - JSON
 - YAML
 - Text
 - ASN.1
 - Ir t.t.



JSON FORMATAS

Tai atviro standarto duomenų formatas, naudojantis atributo/reikšmės poras. Kadangi formatas yra minimalistinis, hierarchinis bei skaitomas, tai jis yra paprastas ir lankstus. Tai yra pats populiausias duomenų apsikeitimo formatas vykdant asinchronines web užklausas (AJAX), kuris didžiąja dalimi pakeitė kitą plačiai naudojamą formatą XML.

Autorius: Douglas Crockford



JSON FORMATAS

- Duomenų tipai:
 - Skaičius: 1, 2, 3 ...
 - Simbolių eilutė: “a”, “AbC”, ...
 - Boolean: true, false
 - Masyvas: [1, 2, 3], [“a”, “AbC”]
 - Objektas: { name: “Student”, age: 25 }
 - Null



JSON FORMATAS

- kelių lygių JSON formato pavyzdys

```
{  
    "firstName": "John",  
    "lastName": "Smith",  
    "isAlive": true,  
    "age": 25,  
    "address": {  
        "streetAddress": "21 2nd Street",  
        "city": "New York",  
        "state": "NY",  
        "postalCode": "10021-3100"  
    },  
    "phoneNumbers": [  
        {  
            "type": "home",  
            "number": "212 555-1234"  
        },  
        {  
            "type": "mobile",  
            "number": "123 456-7890"  
        }  
    ],  
    "children": [],  
    "spouse": null  
}
```



KAS YRA RESTFUL SERVISAI?

- Orientuotas į resursą (angl. Resource based)
- Reprezentacijos (angl. Representations)
- Apribojimai
 - Vienodas interfeisas (angl. Uniform Interface)
 - Be būsenos (angl. Stateless)
 - Klientas-Serveris (angl. Client-Server)
 - Kešuojamas (angl. Cacheable)
 - Sluoksniuota sistema (angl. Layered System)
 - Kodas pagal pageidavimą (angl. Code on Demand)

Originaliai aprašė Roy Fielding savo [disertacijoje](#)



KAS YRA RESTFUL SERVISAI?

- Resursai identifikuojami pagal URI
 - URI formuojamas hierarchijos principu
 - Naudojami daiktavardžiai
- Naudojami HTTP metodai
 - Šie metodai - tai veiksmažodžiai
 - Nusako veiksmą, kuris bus atliekamas su resursu: (GET, POST, PUT, DELETE)
- Rezultato identifikavimui naudojami HTTP statusų kodai



REST-STILIAUS SERVISŲ KŪRIMO GAIRĖS

- Naudoti HTTP veiksmažodžius - tai servisams suteikia daugiau prasmės:
 - GET - Resurso skaitymas (pagal identifikatorių) arba resursų aibės.
 - POST - Resurso sukūrimas.
 - PUT - Resurso modifikavimas (pagal identifikatorių) arba resursų aibės. Gali būti naudojamas resurso sukūrimui, jei identifikatorius yra žinomas iš anksto.
 - DELETE - Resurso trynimas.

Pastaba: GET užklausos neturi pakeisti resurso būsenos.



REST-STILIAUS SERVISŲ KŪRIMO GAIRĖS

- Kiekvienas resursas yra identifikuojamas URI adresu.
Reikia stengtis tokius adresus kurti prasmingus,
išnaudojant hierarchinę struktūrą, kurią mums suteikia
URI sintaksė.
- Vietoj HTTP parametrų naudoti URI identifikatorius
 - OK: /users/123456
 - NOT OK: /api?type=user&id=123456



REST-STILIAUS SERVISŲ KŪRIMO GAIRĖS

- Naudoti URI hierachijos galimybes struktūros išreiškimui (kažkas yra kažko loginė dalis)
- Projektuoti klientams, ne duomenims
- Resursų pavadinimai turėtų būti daiktavardžiai.
Veiksmažodžiai - tai HTTP metodai.
- Atskirus žodžius atskirti vienu iš simbolių: ‘_’ arba ‘-’
- Stengtis minimizuoti adreso ilgj. Jis turi būti maksimaliai aiškus ir trumpas



REST-STILIAUS SERVISŲ KŪRIMO GAIRĖS

- Užklausos rezultatas identifikuojamas naudojant HTTP statusų kodus:

Statusas Aprašymas

200 OK	Bendro pobūdžio sėkmės kodas. Dažniausiai naudojamas statusas.
201 CREATED	Sėkmingas sukūrimo veiksmas (naudojant PUT arba POST metodus).
204 NO CONTENT	Nurodo sėkmingą veiksmo atlikimą, tačiau HTTP atsakymas yra be jokio turinio.
400 BAD REQUEST	Bendrinis klaidos kodas: duomenys nėra validūs, nėra prieinami ir pan.
404 NOT FOUND	Resursas nerastas.



XML IR XHTML

.. IR VARDŲ ERDVĖS



XML

- Panašus į HTML
- Turi žymes - tačiau žymes galime susikurti patys
- XML failo atvaizdavimui naudojamos XSLT transformacijos
- HTML taip pat gali būti XML failas
- Dažniausiai HTML, kuris yra XML failas, vadinamas XHTML ir naudojamas XHTML standartas
- Naudojamas Maven, Tomcat, Spring konfigūracijai



XML

- Galimi įvairūs variantai:

```
<?xml version="1.0" encoding="UTF-8"?>
<saknis/>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<saknis></saknis>
```

```
<saknis/>
```



XML VALIDAVIMAS

- Validavimas kelių lygiu:
 - geras suformavimas (well formedness)
 - <http://www.validome.org/xml/validate/>
 - validacija pagal schema
 - <https://validator.nu/>
- XML taisyklės (žodynas), o ir validavimas - kelių tipu:
 - pagal DTD (Doctype)
 - HTML yra XML, gali būti suvaliduotas pagal DTD
 - pagal XSD (XML Scema)



XML VARDŲ ZONOS

Jeigu norėtume sudaryti dokumentą pagal keletą taisyklių rinkinių (ir vėliau jį validuoti, patikrinti) - pagal HTML ir mūsų žymes:

```
<?xml version="1.0" encoding="UTF-8"?>
<html>
    <manoHtml></manoHtml>
</html>
```



XML VARDŲ ZONOS

O jeigu formuotumėme dokumentą taip, kad turinys būtų body žymėje? Kaip nurodyti, iš kurios vardų zonas?

```
<?xml version="1.0" encoding="UTF-8"?>
<html>
    <body>
        html turinys
        <manoHtml>
            <body>
                mano turinys
            </body>
        </manoHtml>
    </body>
</html>
```



XML VARDŲ ZONOS

Naudojame atributą xmlns be prefikso:

```
<?xml version="1.0" encoding="UTF-8"?>
<html xmlns="http://www.w3.org/TR/html4/">
    <body>
        html turinys
        <manoHtml>
            <body>
                mano turinys
            </body>
        </manoHtml>
    </body>
</html>
```



XML VARDŲ ZONOS

Pridedame ir asmeninę vardų zoną kitu prefiksui:

```
<?xml version="1.0" encoding="UTF-8"?>
<html xmlns="http://www.w3.org/TR/html4/"
      xmlns:prefiksas="http://mano.lt">
<body>
    html turinys
    <prefiksas:manoHtml>
        <prefiksas:body>
            mano turinys
            </prefiksas:body>
        </prefiksas:manoHtml>
    </body>
</html>
```



PRADINIS PUSLAPIO KODAS

```
<HTML>
    <HEAD>
        <TITLE>Infobalt mokykla</TITLE>
    </HEAD>
    <BODY>
        Rodomas tekstas.
    </BODY>
</HTML>
```



PRADINIS PUSLAPIO KODAS

```
<HTML xmlns="http://www.w3.org/TR/html5/">
    <HEAD>
        <TITLE>Infobalt mokykla</TITLE>
    </HEAD>
    <BODY>
        Rodomas tekstas.
    </BODY>
</HTML>
```



PRADINIS PUSLAPIO KODAS JSF

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html">
    <h:head>
        <title>Infobalt mokykla</title>
    </h:head>
    <h:body>
        Rodomas tekstas.
    </h:body>
</html>
```



SVG VARDŲ ZONOS PAVYZDYS

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>SVG</title>
    <meta charset="UTF-8" />
  </head>
  <body>
    <svg xmlns="http://www.w3.org/2000/svg">
      <rect stroke="black" fill="blue" x="45px" y="45px"
            width="200px" height="100px" stroke-width="2" />
    </svg>
  </body>
</html>
```



UŽDUOTIS 1 - XML

- Pabandykite kiekvieną iš anksčiau rodytų pavyzdžių
 - patikrinkite kodo well formness
 - patikrinkite validaciją pagal schemą



KARKASAI

KOMPONENTŲ KARKASAI



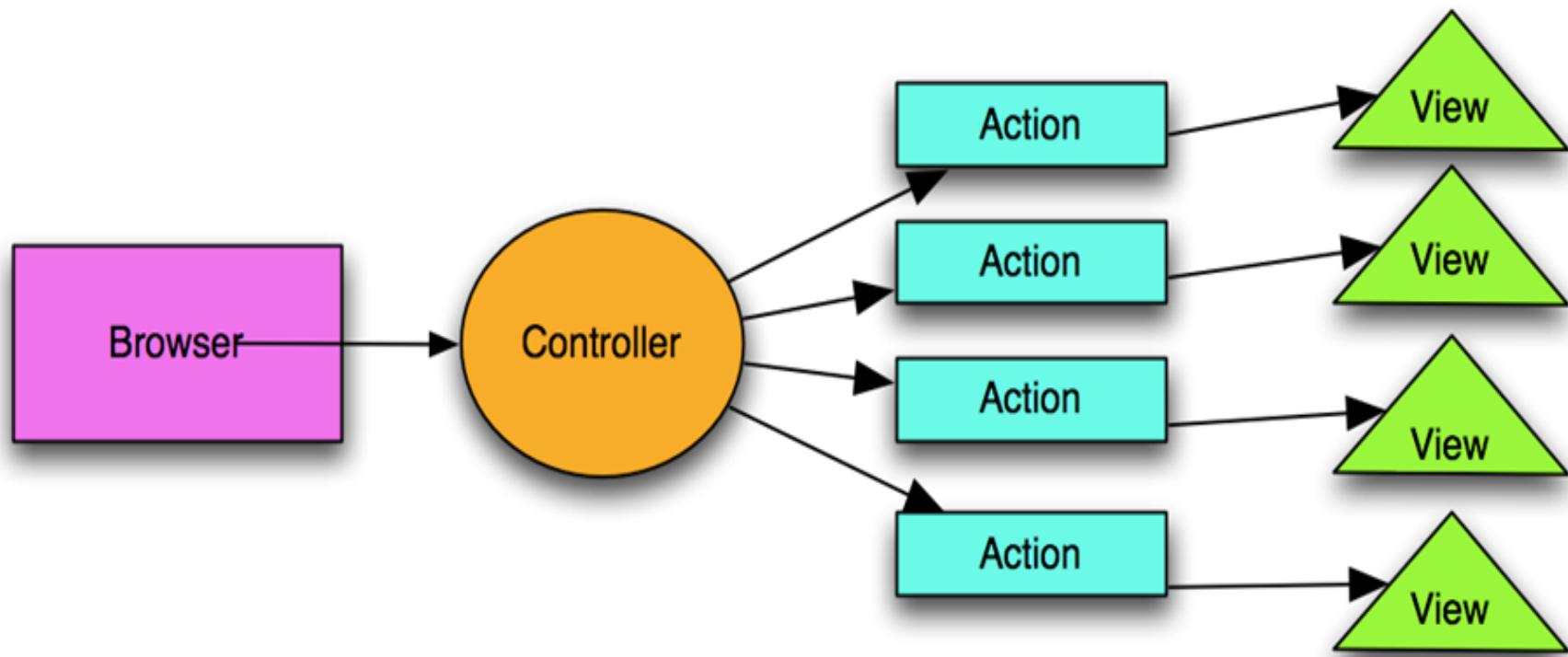
AKADEMIJA.LT
INFOBALT IR TECH CITY

KARKASAI

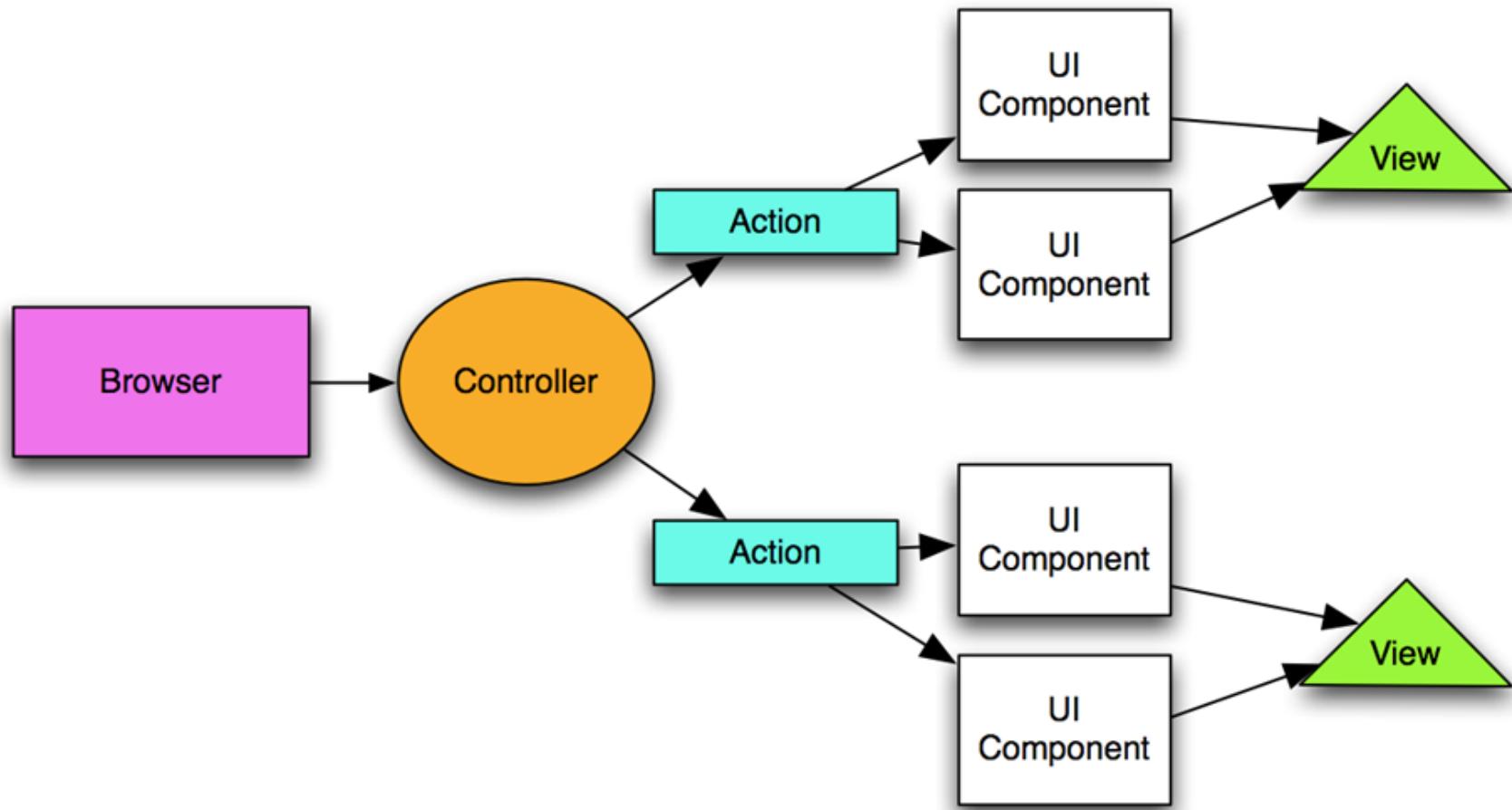
- Veiksmų karkasai (action)
 - Spring MVC, Struts, Struts2, Rails
- Hibridiniai karkasai (hybrid)
 - Tapestry, Wicket
- UI komponentų karkasai
 - JSF, Rife, Echo2
 - Dauguma UI JS karkasų: ReactJS, AngularJS ir kt.



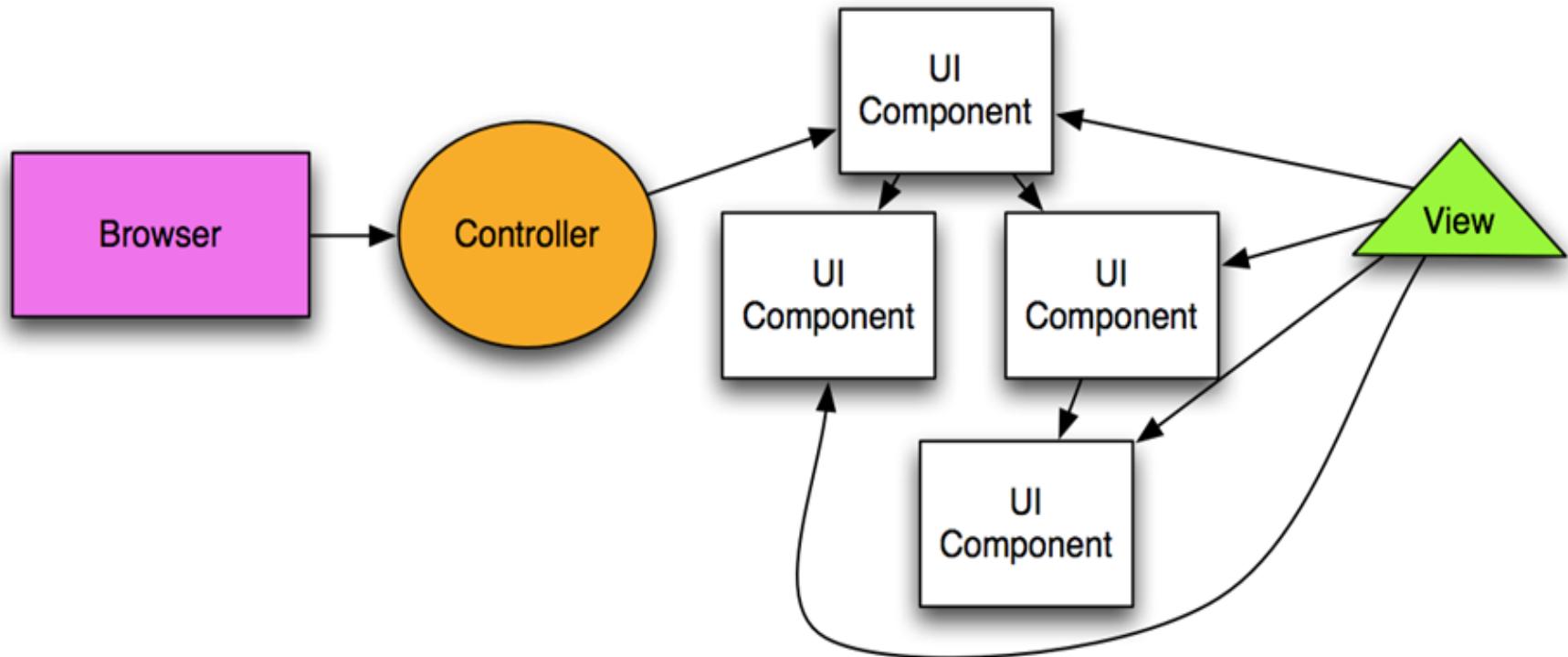
VEIKSMŲ KARKASAI



HIBRIDINIAI KARKASAI



UI KOMPONENTŲ KARKASAI



KOMPONENTŲ KARKASAI

- Karkasas su komponentais, kurie yra vaizde, komunikuojant tiesiogiai
- Egzistuoja komponentų medis, todėl lengva eiti per komponentus ir valdyti jų gyvavimo ciklą
- Saugoma komponentų būsena (state)
- Modelis dinamiškai atsinaujina pagal komponentus, esančius vaizde
- Gerai tinka sudėtingesniems puslapiams, kuriuose reikalingas didelis komponentų skaičius



DAUGIASLUOKSNĖ SISTEMŲ ARCHITEKTŪRA



SISTEMŲ ARCHITEKTŪRA

- IT sistemų architektūra yra aukšto lygmens dizainas, kuriuo yra paremta sistema.
- Architektūra turi sekančias savybes:
 - komponentai,
 - bendradarbiavimas (kaip komponentai saveikauja),
 - jungtys (kaip komponentai bendrauja).



POPULIARIAUSIOS ARCHITEKTROS

- Kliento-serverio,
- Sluoksniuota (angl. tiered),
- Lygiarangių (angl. peer-to-peer),
- Kitos (pvz pipes and filters).



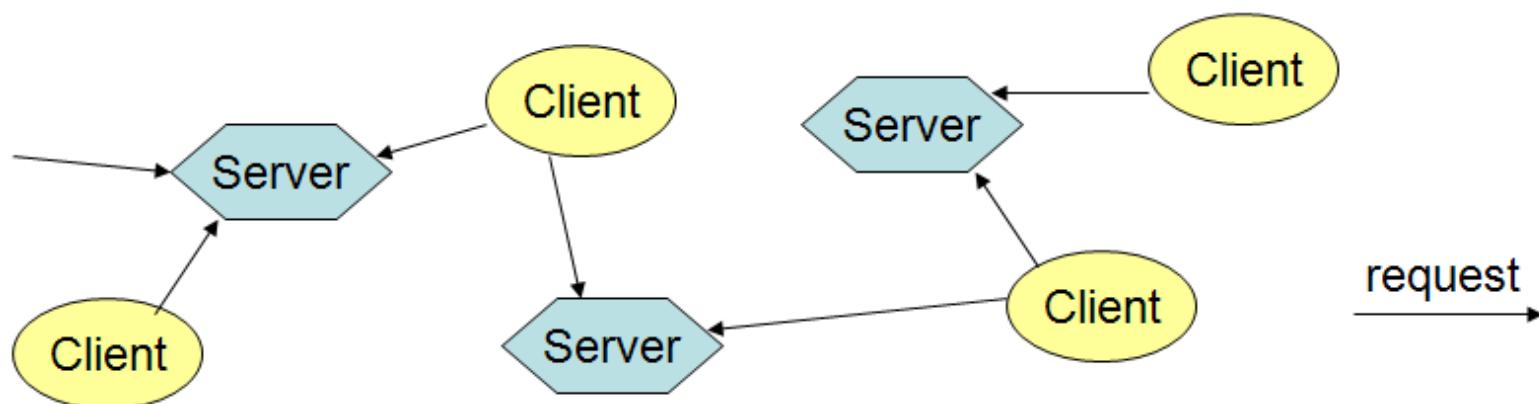
KLIENTO-SERVERIO ARCHITEKTŪRA

- Kiekvienas kliento-serverio architektūros komponentas turi arba kliento, arba serverio rolę:
 - Klientas - komponentas kuris sukūria užklausas. Klientai yra aktyvus ir inicijuoja tranzakcijas.
 - Serveris - komponentas kuris patenkina užklausas. Serveriai yra pasyvus ir reguoja į kliento užklausas.
 - Serveris gali aptarnauti daugiau nei vieną klientą.
- Žiniatinklis yra kliento-serverio sistema.



Kliento-serverio architektūra

- Interneto naršyklės elgiasi kaip klientai ir sukuria užklausas žiniatinklio serveriams.
- Žiniatinklio serveriai atsako į užklausas ir / arba atlieka skaičiavimus.



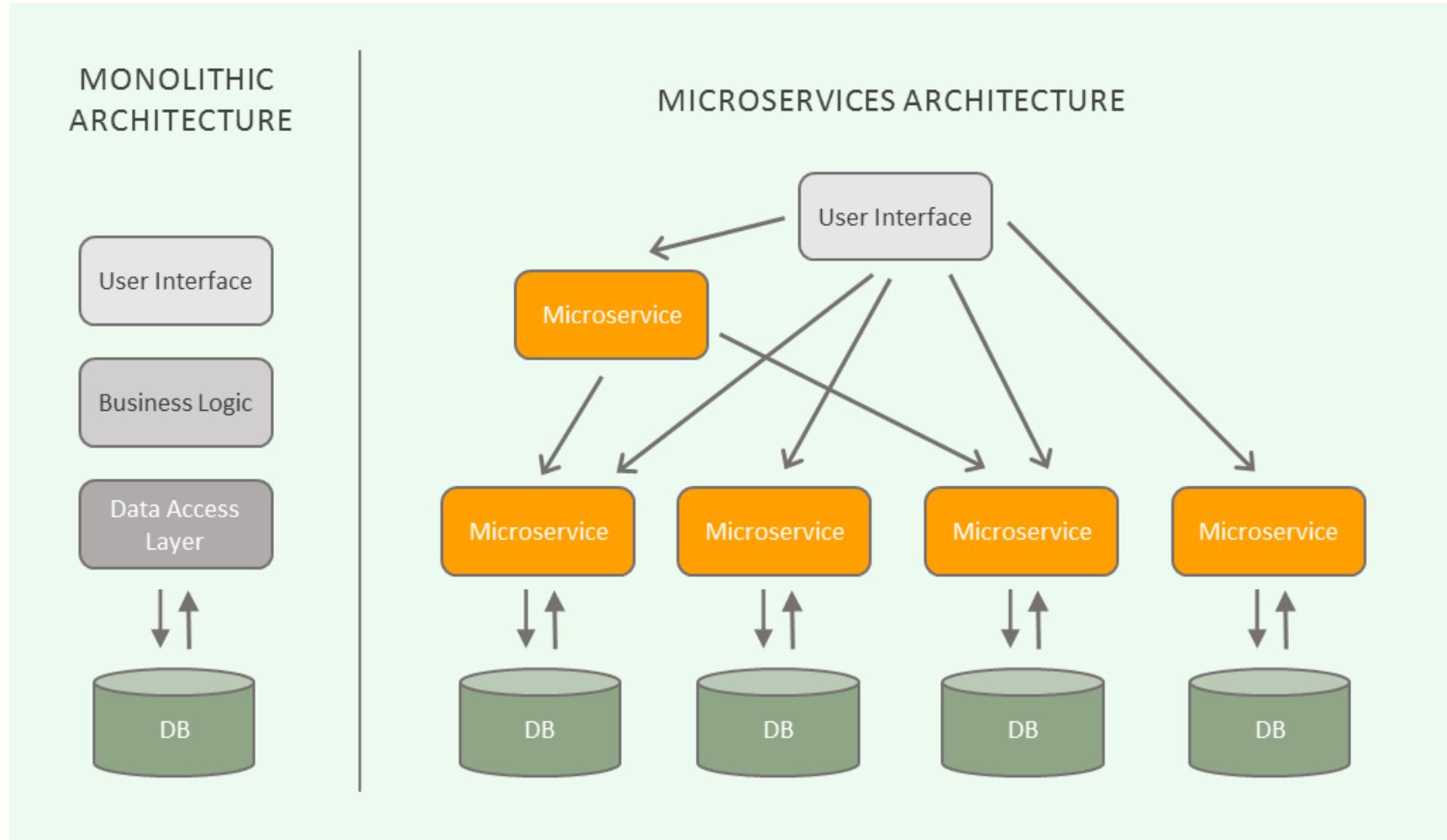
MIKROSERVIAI

In short, the microservice architectural style is an approach to developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms, often an HTTP resource API. These services are built around business capabilities and independently deployable by fully automated deployment machinery. There is a bare minimum of centralized management of these services, which may be written in different programming languages and use different data storage technologies. -- James Lewis and Martin Fowler

- pradėjo populiarėti nuo 2014
 - <https://www.martinfowler.com/microservices/>
 - <https://martinfowler.com/articles/microservices.html>



MIKROSERVISAI IR MONOLITINĖ SISTEMA



CENTRALIZUOTAS / PASKIRSTYTAS APDOROJIMAS

- Kliento-serverio architektūrą galima laikyti viduriu tarp:
 - Centralizuoto apdorojimo - skaičiavimai atliekami naudojant centrinę platformą, kuri pasiekiamas per “kvailus” terminalus.
 - Paskirstyto apdorojimo - skaičiavimai atliekami taip pat ir ant naudotojo platformos.



STORAS / PLONAS KLIENTAS

- Storas klientas:
 - Dalis aplikacijos veikia kliento pusėje.
 - Klientas žino kaip struktūrizuoti duomenis ir kur jie yra.
 - Skirtingi klientai gali naudoti aplikaciją skirtingais būdais.
- Plonas klientas:
 - Klientas yra mažiau sudėtingas.
 - Dauguma kodo veikia serverio pusėje.
 - Tinklo saveika tarp kliento ir serverio yra sumažinta.



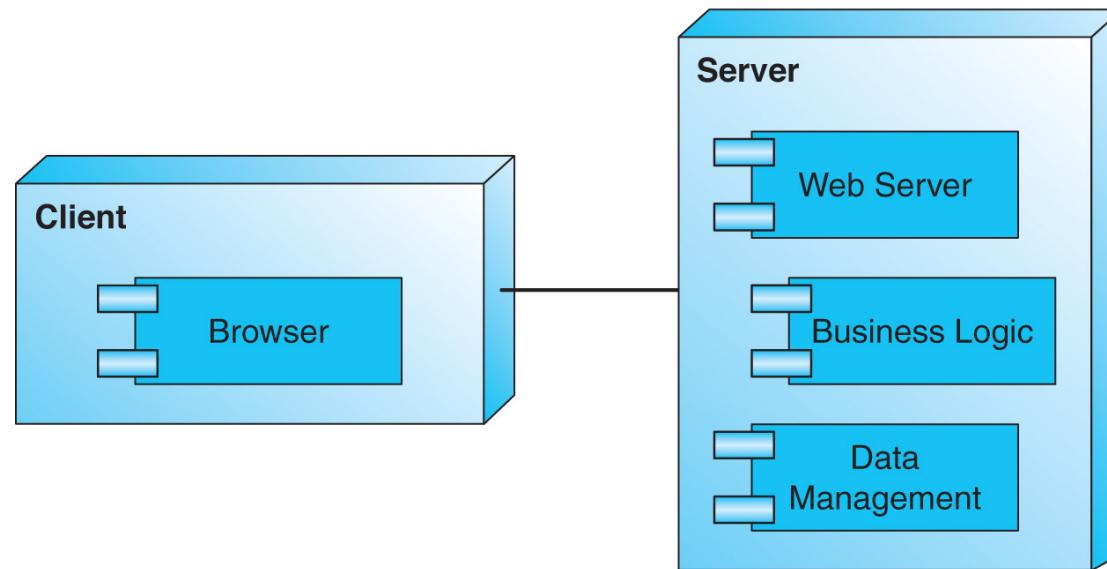
SLUOKSNIUOTA ARCHITEKTŪRA

- 1 sluoksnio architektūra:
 - monolitinės IT sistemos,
 - naudotojo sąsaja, veiklos logika ir duomenų valdymas yra sujungti į vieną sluoksnį.
- Žiniatinklio aplikacijos įprastai realizuojamos naudojant 2, 3 arba daugėlio (N) sluoksniių architektūrą.
- Kiekvienas sluoksnis turi unikalią atsakomybę.



2 SLUOKSNIŲ ARCHITEKTŪRA

- 1 sluoksnis yra naudotojo sasajos sluoksnis.
- 2 sluoksnis yra veiklos logikos ir duomenų valdymo sluoksnis.



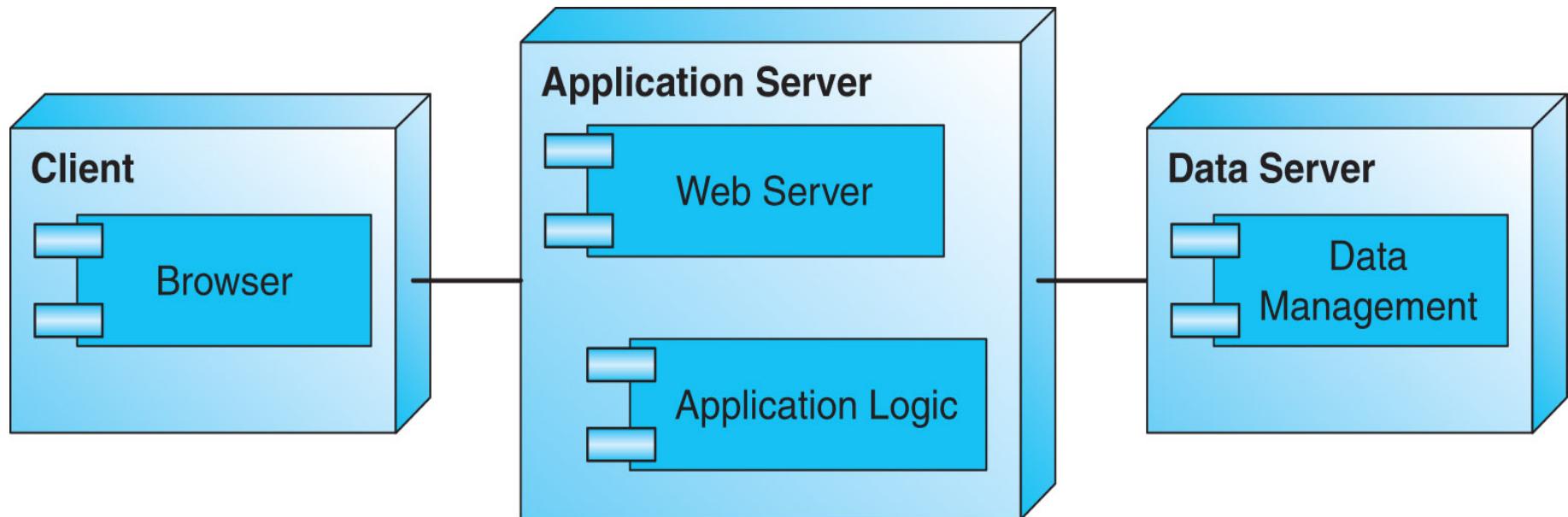
2 SLUOKSNIŲ ARCHITEKTŪROS CHARAKTERISTIKOS

- Privalumai: nebrangi
- Trūkumai:
 - komponentų tarpusavio priklausomybė
 - nėra dubliavimo (angl. redundancy)
 - ribotas išplečiamumas (angl scalability)
- Tipinė aplikacija:
 - 10-100 naudotojų
 - mažos įmonės arba organizacijos



3 SLUOKSNIŲ ARCHITEKTŪRA

3 sluoksnis paprastai perima iš 2 sluoksnio duomenų valdymo atsakomybę.



3 SLUOKSNIŲ ARCHITEKTŪROS CHARAKTERISTIKOS

- Privalumai:
 - padidintas našumas dėl specializuotos aparatinės įrangos
 - sumažinta komponentų tarpusavio priklausomybė
 - padidintos išplečiamumo galimybės
- Trūkumai: nėra dubliavimo (angl. redundancy)
- Tipinė aplikacija:
 - 100-1000 naudotojų
 - vidutinės įmonės arba regiono lygio organizacijos



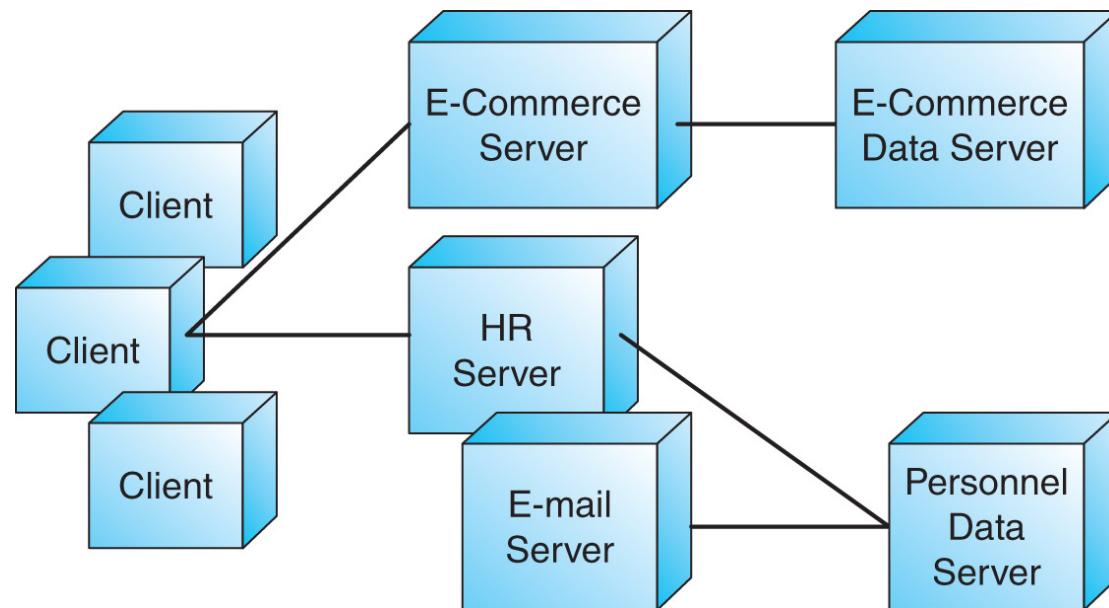
DAUGELIO SLUOKSNIŲ ARCHITEKTŪRA

- Daugėlio (N) sluoksnių architektūra išplečia 3 sluoksnių architekturą vienu ar keleta galimų būdų:
 - sluoksnio funkcionalumo replikavimas
 - specializuota funkcija sluoksnuje
 - portalo paslaugos, orientuotos į žiniatinklio apkrovos valdymą



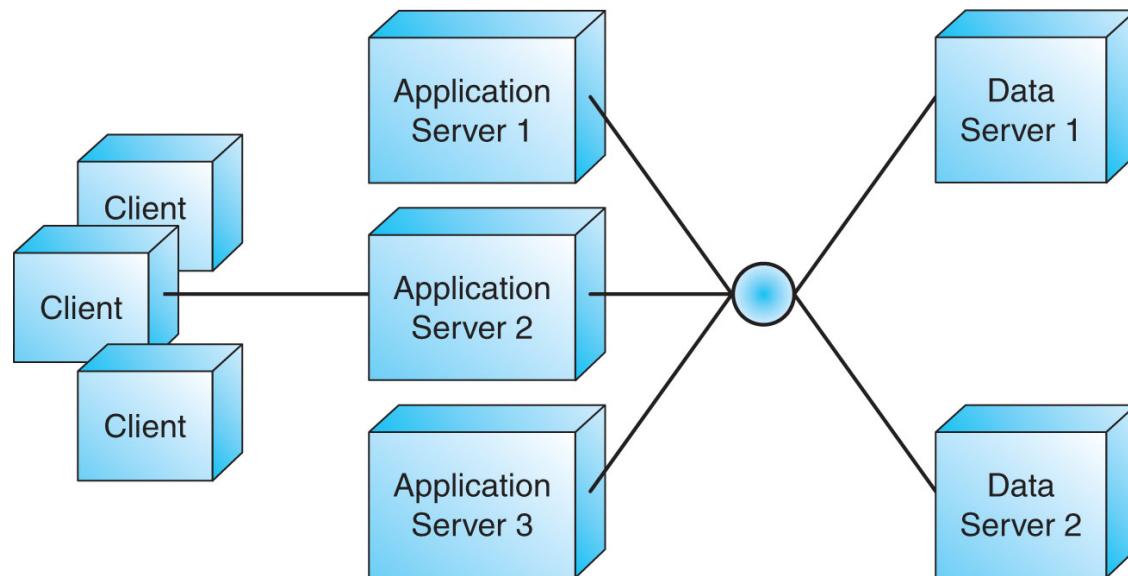
REPLIKAVIMAS

Aplikacijų ir duomenų serveriai yra replikuoti ir bendrai dalinasi apkrova.



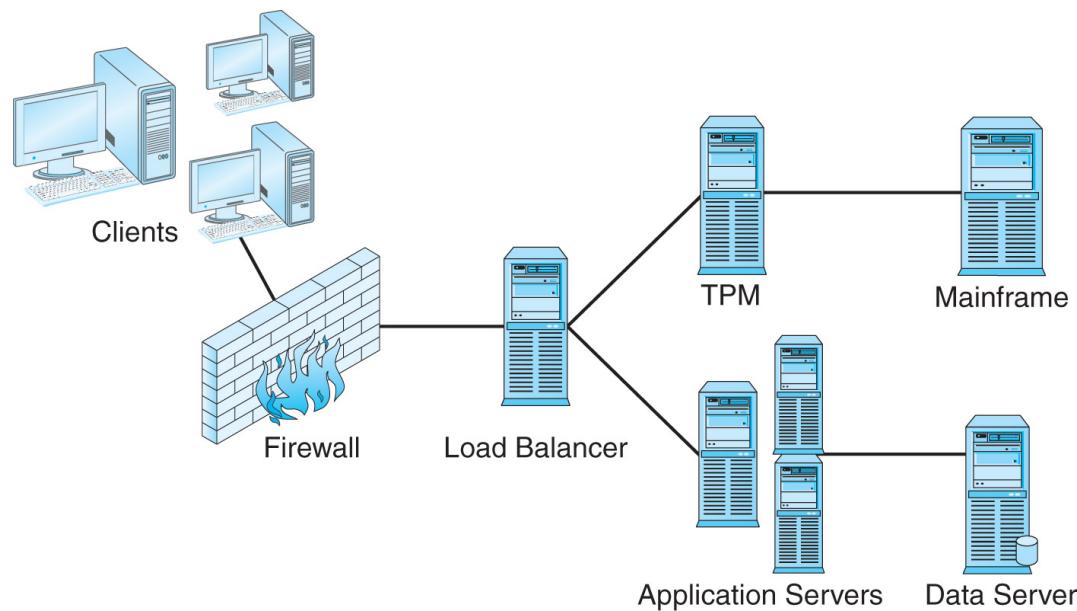
SPECIALIZACIJA

Serveriai yra specializuoti ir pagal specifinę funkciją, apdoroja jiems paskirtą krūvį.



PORTALO PASLAUGOS

Portalo serveriai apdoroja įeinantį srautą, sumažindami aplikacijų serverio apkrovą.

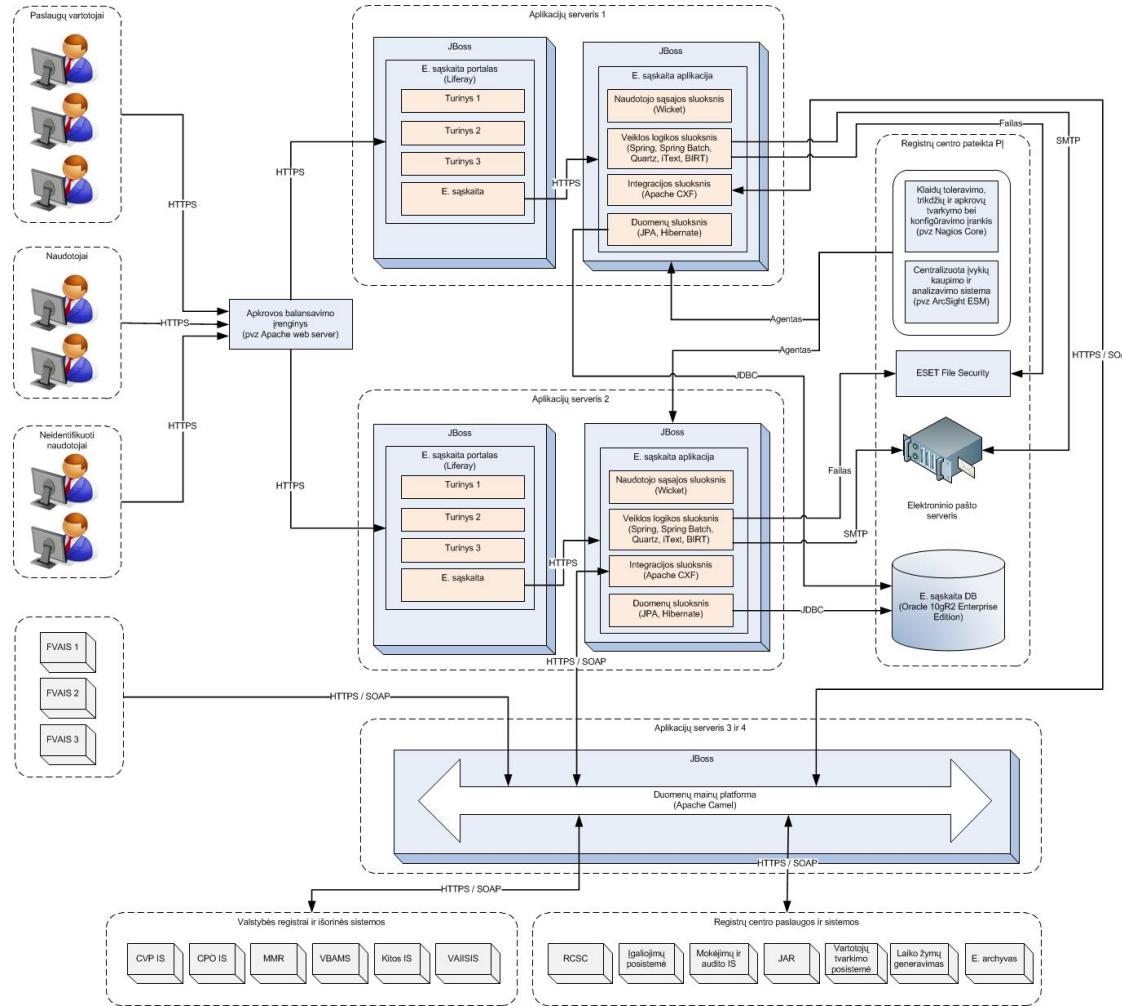


N SLUOKSNIŲ ARCHITEKTŪROS CHARAKTERISTIKOS

- Privalumai:
 - atsieti programinės įrangos komponentai
 - lankstus komponentų pridėjimas / pašalinimas atsižvelgiant į apkrovą
 - išplečiamumas
 - dubliavimas
- Trūkumai: didesni priežiūros kaštai
- Tipinė aplikacija:
 - 1000+ naudotojų
 - didelės įmonės arba organizacijos



N SLUOKSNIŲ ARCHITEKTŪROS PAVYZDYS



KOKIĄ APLIKACIJĄ KURSIME

- Daugiasluoksnę verslo aplikaciją:
 - naudotojo sąsajos sluoksnis - React biblioteka-karkasas
 - veiklos logikos sluoksnis - Spring karkasas
 - duomenų valdymo sluoksnis - JPA (Java Persistence API)
- Aplikacijos į mikro servisus neskaidysime
 - tačiau analogiškai būtų kuriami ir mikro servisi



UŽDUOTIS 2 - PABAIGTI APLIKACIJĄ

- Pabaigti parduotuvės aplikaciją
 - dalis funkcionalumo čia
<https://itpro2017.herokuapp.com/>
 - REST API čia
<https://itpro2017.herokuapp.com/swagger-ui.html#/>



UŽDUOTIS 2 - PABAIGTI APLIKACIJĄ

- UI turi būti galima:
 - pirmame puslapyje matyti navigacijos eilutę ir produktų sąrašą
 - turi būti galima įvesti naudotojo vardą ir aplikacija su juo turi dirbti
 - produktų administravimo lange pridėti, redaguoti, ištrinti produktą
 - įdėti produktą į nurodyto naudotojo krepšelį
 - peržiūrėti krepšelį
 - ištrinti produktą iš krepšelio



KITOJE PASKAITOJE

Tomcat. Maven. Java. Spring

