Tīmekļa risinājums ar CodeIgniter - pamati

Tīmekļa tehnoloģijas II, 2011

Krišs Rauhvargers

PROJEKTĒŠANA

Apstrādājamie dati

- No biznesa prasību apraksta var secināt, *kas* būs tie dati, ko apstrādās.
  - cilvēku vārdi, telefona numuri, receptes, riepu izmēri, paroles...
- Ar IS palīdzību parasti vajadzēs kaut ko darīt ar šiem datiem, piemēram,
  - nolasīt no datu glabātuves (datubāzes)
  - saglabāt datubāzē

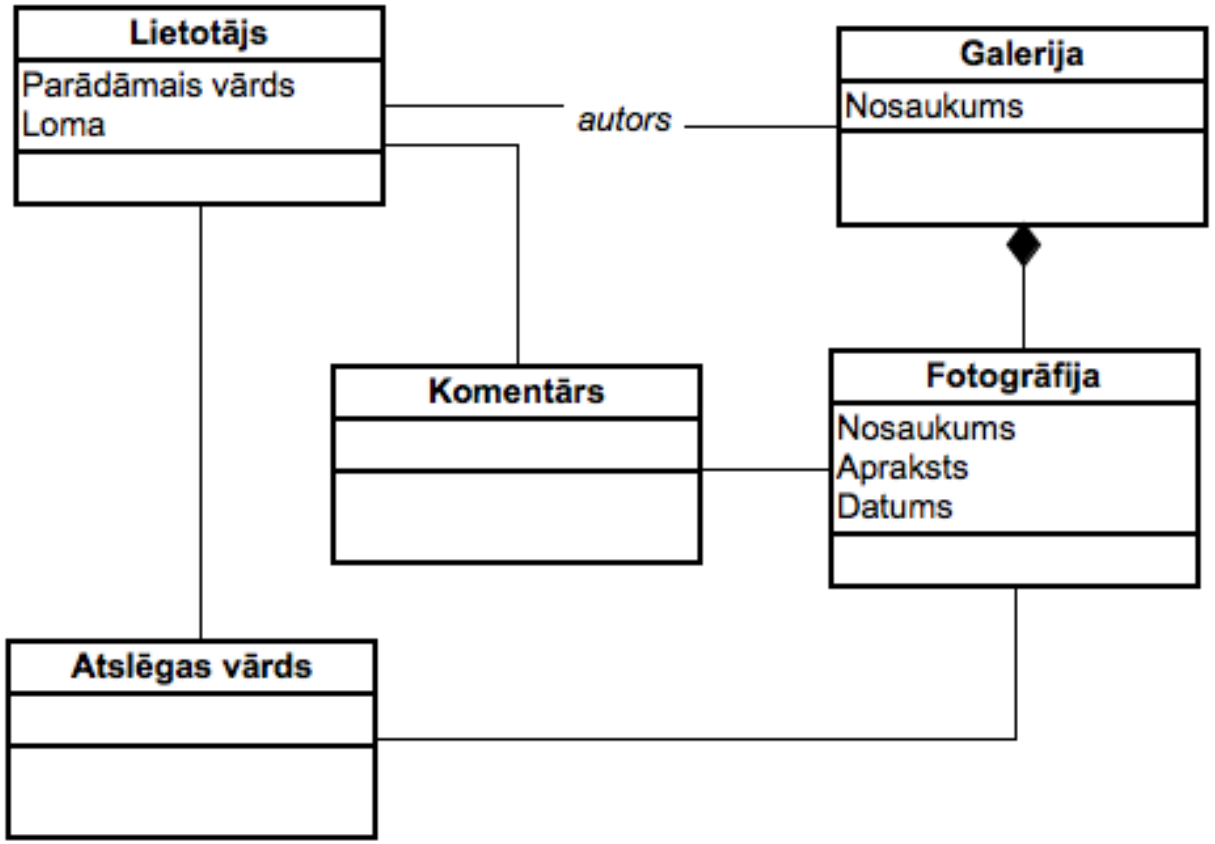
Biznesa jēdzieni > kontrolleri

- Katrs "biznesa jēdziens" (resurss) potenciāli var kļūt par kontrolleri MVC risinājumā.
- Atbilstība ne obligāti būs 1:1
  - piemēram, "fotogrāfija": var nodalīt attēla aprakstu no paša attēla faila
  - daži jēdzieni var realizēties kā darbības ar citiem

Resursi MVC modelī

- Biznesa jēdziens (resurss) -> kontrolleris
- Darbības, ko veikt ar resursu -> kontrollera metodes
- Katram kontrollerim ir sava URL adrese
  - `http://example.com/photo/`
- Metodes pakārtojas zem kontrollera adreses
  - `http://example.com/photo/edit/`
- Parasti metodei vajadzīgs resursa identifikators
  - `http://example.com/photo/edit/941`

Fotogalerijas modelis



Fotogalerijas kontrolleri (daži)

- Lietotājs `http://example.com/user/`
  - autentificēties (ievada paroli, sistēma pārbauda)
    - `http://example.com/user/`**`login`**
  - rediģēt savus datus (vārdu, uzvārdu..)
    - `http://example.com/user/edit/411`
- Galerija `http://example.com/gallery/`
  - skatīt (parāda visas fotogrāfijas šajā galerijā)
    - `http://example.com/gallery/view/102`
  - izveidot (galerijas nosaukums, pieejamība)
    - `http://example.com/gallery/create/`
  - labot aprakstu (galerijas nosaukums, pieejamība)
    - `http://example.com/gallery/edit/491`

Fotogalerijas kontrolleri (cont)

- Fotogrāfija `http://example.com/photo`
  - veidot jaunu (nosaukums, kurā galerijā)
    - `http://example.com/photo/create/102`
  - skatīt (rāda aprakstu un fotogrāfijas failu)
    - `http://example.com/photo/view/419`
  - labot (nosaukums, galerija)
    - `http://example.com/photo/edit/419`
- Fotogrāfijas fails `http://example.com/image/`
  - augšupielādēt (piesaistīts kādai fotogrāfijai)
    - `http://example.com/image/upload/419`
  - skatīt (atgriež jpg failu, parametrs - izmērs)
    - `http://example.com/image/view/450/`

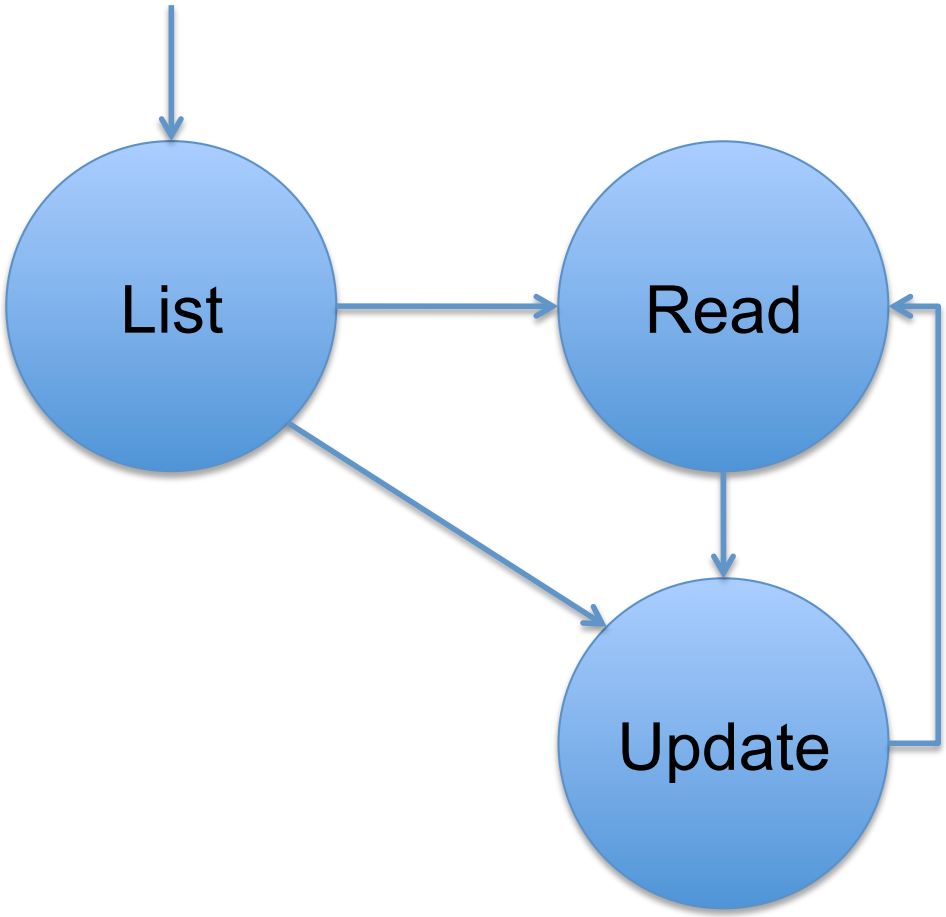
CRUD

- Pilnvērtīgam datu apstrādes modelim jāsatur visas šīs darbības
  - Jauna ieraksta veidošana (**C**reate)
  - Ieraksta datu nolasīšana (**R**ead)
  - Ieraksta labošana (**U**pdate)
  - Ieraksta dzēšana (**D**elete)
- Tās atbalstītas visur: SQL relāciju datubāzēs, failu sistēmā, FTP protokolā...
- Lai sistēma būtu praktiski lietojama, vajadzīgas vēl vismaz divas operācijas:
  - Datu saraksta nolasīšana (List)
  - Meklēšana datu sarakstā (Search)

Lietojuma scenārijs

- Apskatu galeriju sarakstu (Gallery>List)
- Uzreiz neredzu vajadzīgo galeriju, meklēju pēc nosaukuma (Gallery>Search)
- Ieraugu galeriju, klikšķinu "Skatīt" (Gallery>Read)
- Pamanu, ka virsrakstā kļūda, to laboju (Gallery>Edit)

Lietojuma scenārijs kā galīgs automāts



"Bumbas" ir operācijas, kas tiek veiktas ar objektu.

Bultiņas parāda operācijas, ko lietotājs pašreizējā stāvoklī var veikt tālāk.

Stāvokļi un tīmekļa formas?

- Tīmekļa arhitektūra prasa ieviest papildus stāvokļus
  - Klients: Gribu pievienot jaunu dokumentu
  - Serveris: Ģenerē jauna ieraksta veidošanas formu
  - Klients: sagatavo datus un sūta uz serveri
  - Serveris: nepareizi aizpildīti dati laukā A. Ģenerē formu vēlreiz, izceļ nepareizos laukus
  - Klients: labo/papildina datus un sūta uz serveri
  - Serveris: Saglaba datus.

CRUD operāciju apskats

ELEMENTA NOLASĪŠANA

Elementa nolasīšanas lapa

- URL adrese norāda
  - resursa veidu
  - identificē konkrēto nolasāmo ierakstu
  - Piemēram
    - http://mansdators/phones/view/1
    - http://mansdators/read_phone.php?id=1
- Identificēšanai parasti izmanto DB primāro atslēgu, bet var arī citādi, piem:
  - datums+numurs pēc kārtas attiecīgajā datumā
    - http://wp.com/2010/09/20/1/
  - datums+virsraksta fragments
    - http://wp2.com/2010/09/20/Hello_World

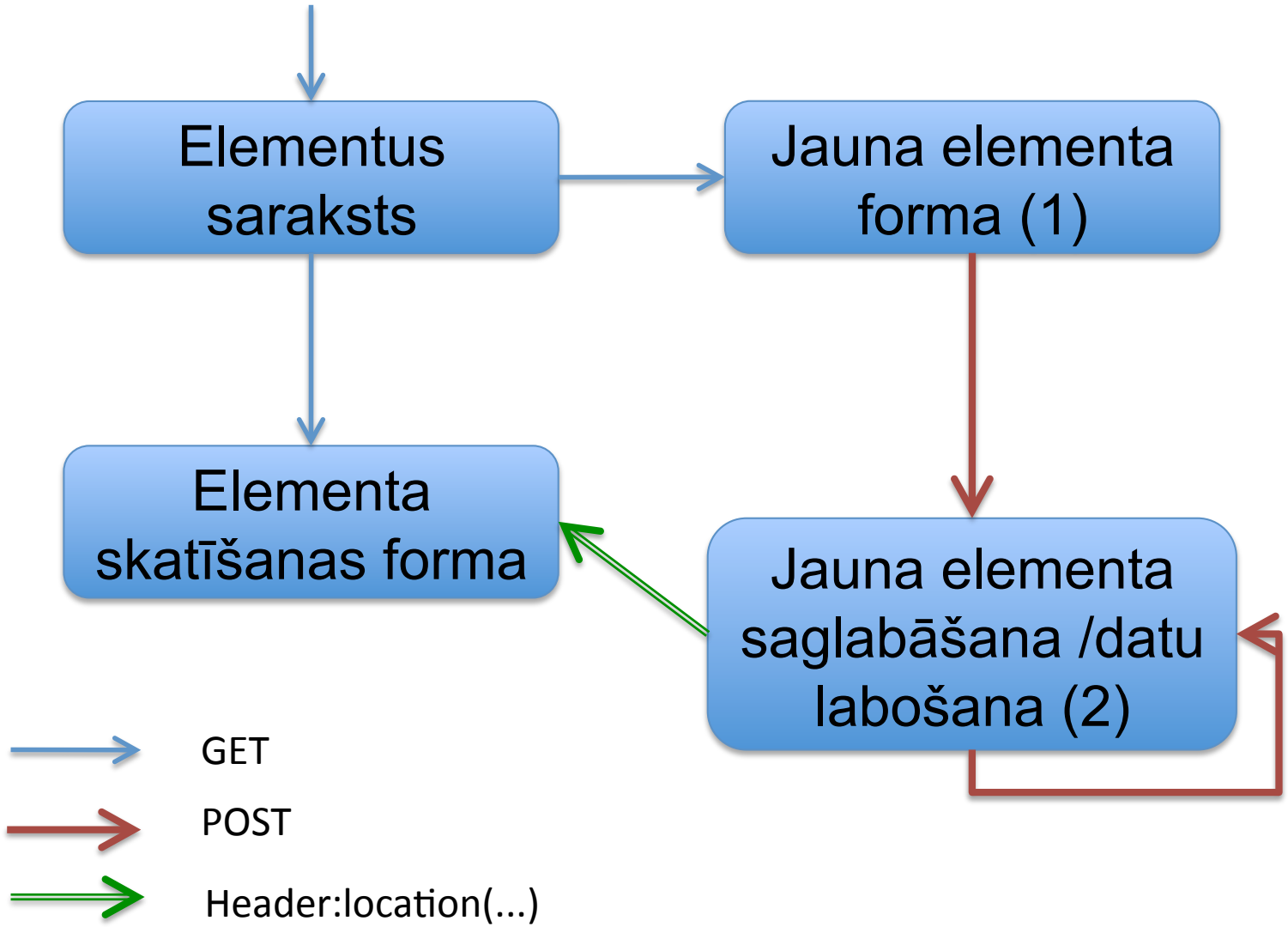
Elementa nolasīšanas lapa (turp.)

- Veicamie uzdevumi:
  - lietotāja autentifikācija + autorizācija
    - kas ir šis lietotājs
    - vai šis lietotājs drīkst skatīt šo ierakstu
  - ieraksta meklēšana
  - ja ieraksts nav atrasts
    - pāradresācija uz kļūdas lapu
  - ja ieraksts atrasts
    - skatīšanas formas sagatavošana

CRUD operāciju apskats

JAUNU ELEMENTU VEIDOŠANA

Jauna ieraksta veidošana

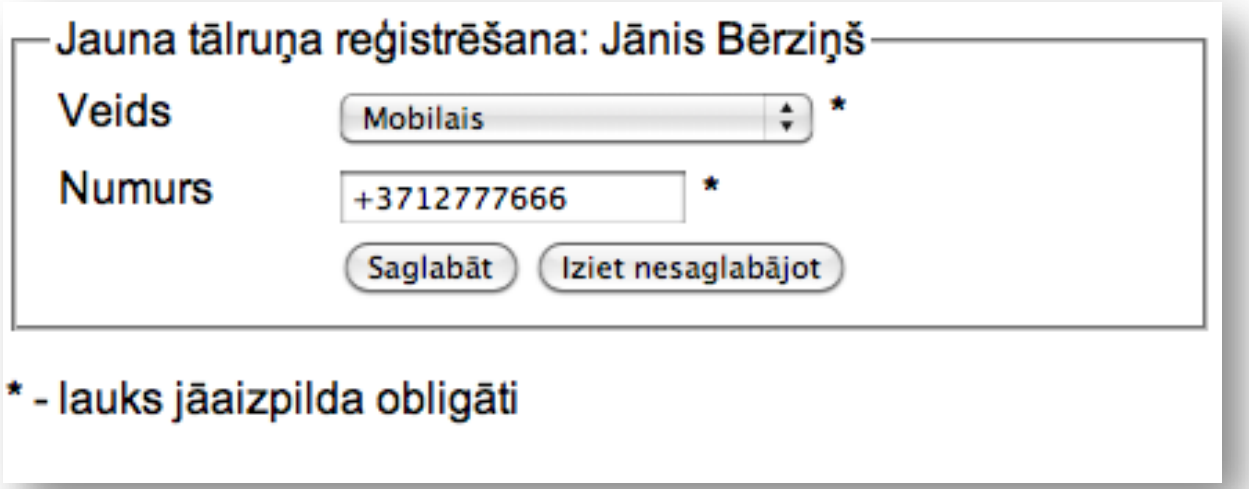


Jauna elementa forma

- Formas URL jāiekļauj:
  - kāds elements tiks veidots
  - (ja nepieciešams) saistīto elementu dati
  - piemēram:
    - http://mansdators/phones/new/Jānis
    - http://mansdators/new_phone.php?person=123
- Formu var ielādēt pēc patikas daudz reižu
- Saistītie elementi
  - Izmanto, lai veidotu sasaistes DB līmenī
  - Var lietotājam parādīt, ka veidos saiti
  - Ja nolasa saistītos datus, jāpārliecinās, vai lietotājam ir tādas tiesības!

Jauna elementa forma (turp.)

- Lietotājam saprotamā veidā jāizskaidro, kas kurā lauciņā jāraksta
- Vizuāli jānodala tie lauki, kuru izmantošana obligāta
- Jācenšas lietot klienta puses pārbaudes (validāciju)
- Noderēs "Aizvērt/Iziet/Atcelt" poga



Jauna elementa forma (turp.)

- Datus tālāk sūta POST protokolā
- Informāciju, kas saņemta kā parametri, var iekodēt `<input type='hidden' />` laukos

Jauna elementa saglabāšanas lapa

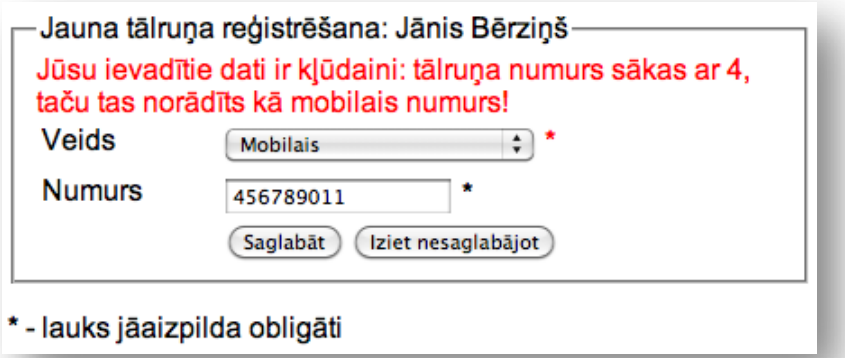
- Saglabāšanu veic tas pats kontrolleris, kur ir jauna elementa forma.
- Tā var būt tā pati forma, kas iepriekšējā solī
  - Kontrolleris pārbauda, vai saņemti POST dati un rīkojas
- Var būt arī atsevišķa forma

Jauna elementa saglabāšanas lapa (2) (turp.)

- Veicamie darbi:
  - autorizēšana:
    - kas ir šis lietotājs
    - vai šis lietotājs drīkst pievienot jaunus ierakstus
  - datu kvalitātes pārbaude
    - vai aizpildīti visi lauki
    - vai vērtības iekļaujas intervālos
  - Ja dati OK
    - saglabāšana
    - pāradresācija (*header:location*) uz jaunizveidotā ieraksta apskates lapu
  - Ja datos ir problēmas
    - Problēmas skaidrojuma sagatavošana
    - Atkal parāda formu, tajā saglabājot lietotāja datus

Elementa saglabāšanas lapa: kļūdu labošana

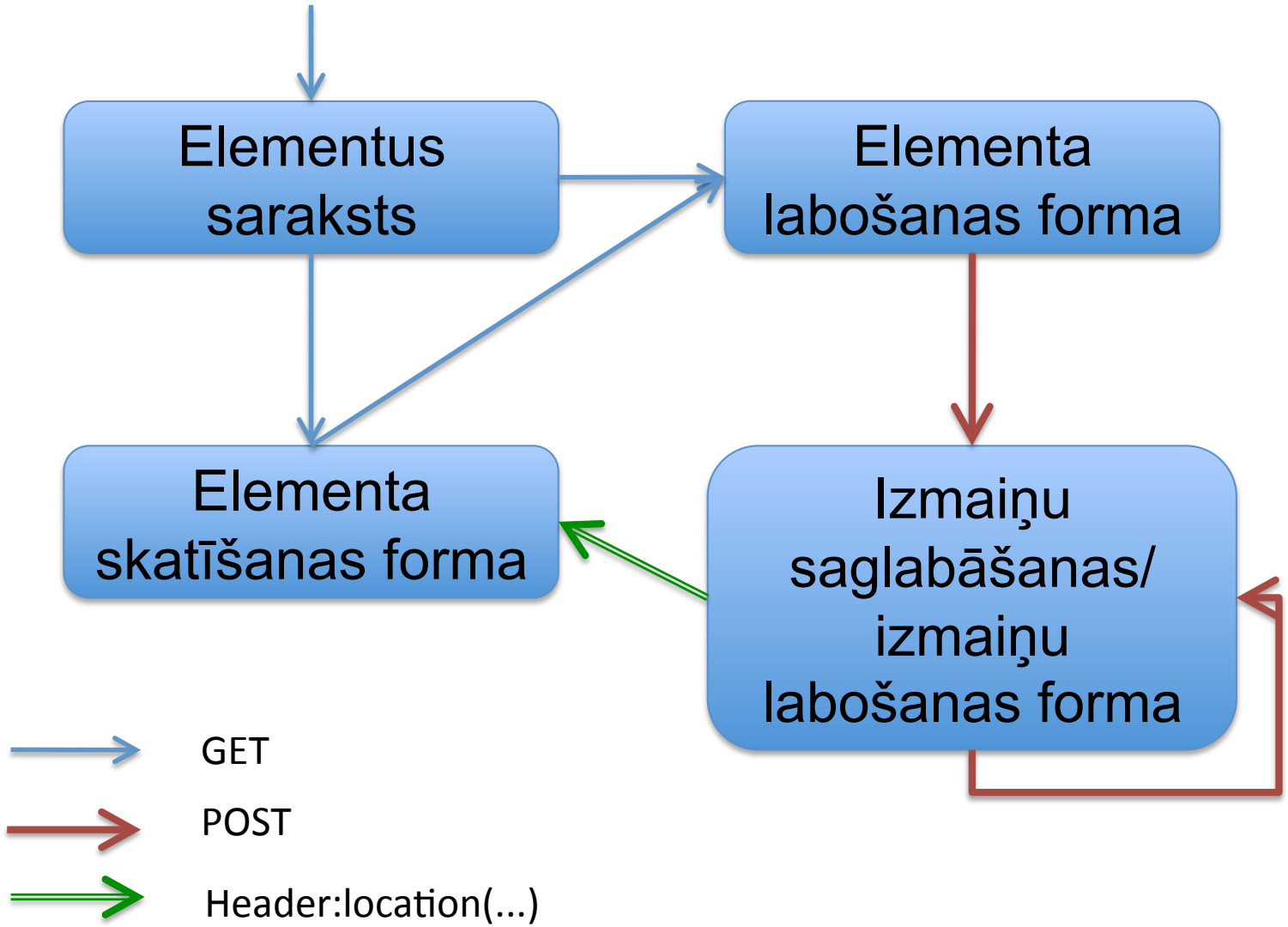
- Ja formā ir kļūdas, tās izceļ labi redzamas.
- Nav vēlams "pazaudēt" lietotāja datus
- Formas datus atkal sūta uz saglabāšanas lapu



CRUD operāciju apraksts

ELEMENTA LABOŠANA

Elementa labošanas soļi



Elementa labošanas forma

- URL adresē norāda *resursa veidu* + *identificē ierakstu* (identiski kā nolasīšanas formā)
- Sagatavojot formu, ņem pašreizējos datus no datu glabātuves
- Autorizējot:
  - vai lietotājs drīkst pašlaik *lasīt* ierakstu
  - vai drīkstēs *mainīt* ierakstu
- Var būt gadījumi, kad daļa informācijas nav maināma

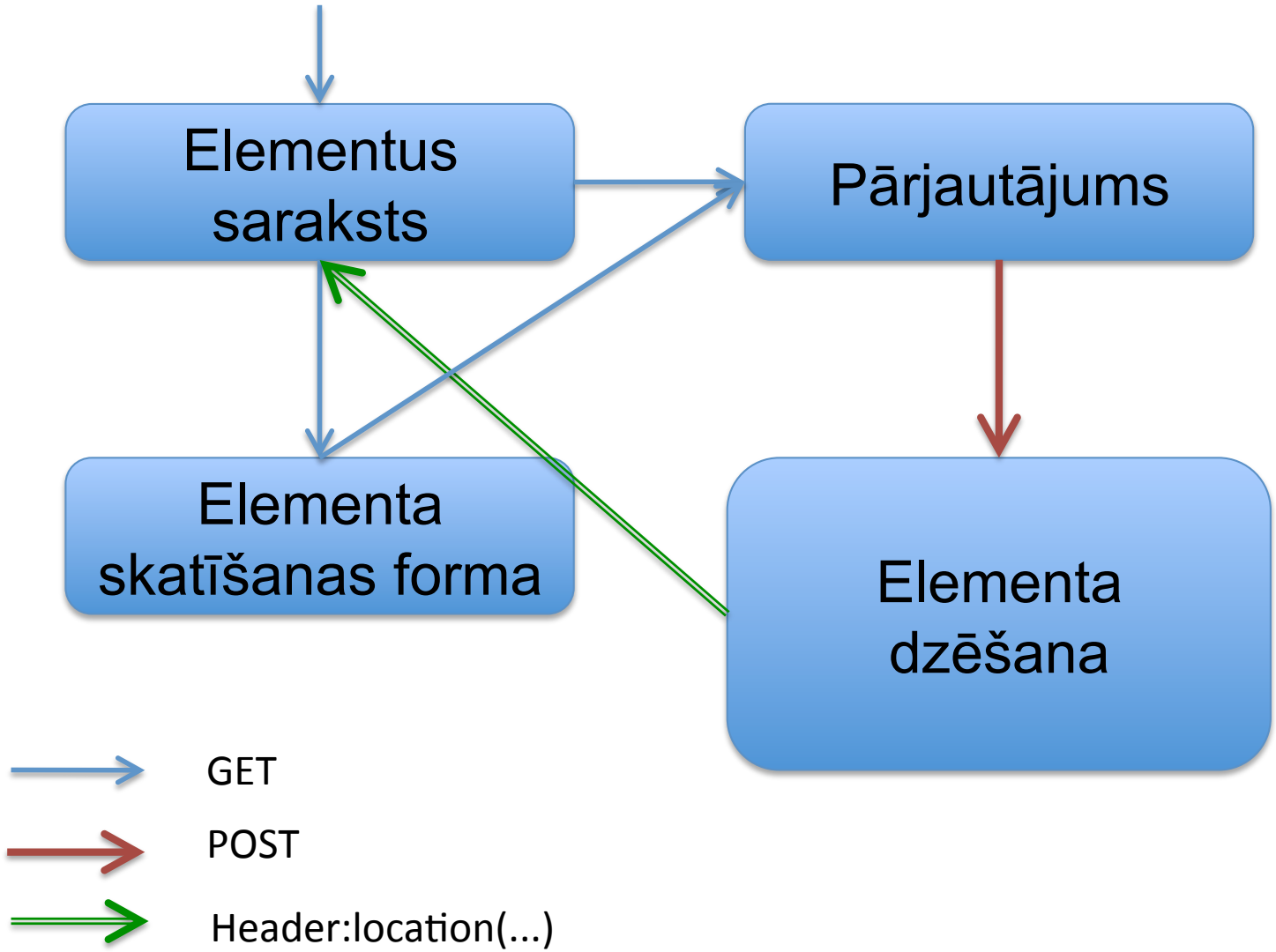
Izmaiņu saglabāšanas forma

- URL norāda resursa veidu, pārējie dati kā POST mainīgie
- Atkārtoti jāveic lietotāja autentifikācija/ autorizācija
- Ja datos ir kļūdas
  - rāda labošanas formu
  - formas laukos lietotāja sūtītā informācija, nevis datu glabātuvē esošās vērtības

CRUD operāciju apskats

ELEMENTA DZĒŠANA

Elementa dzēšanas soļi



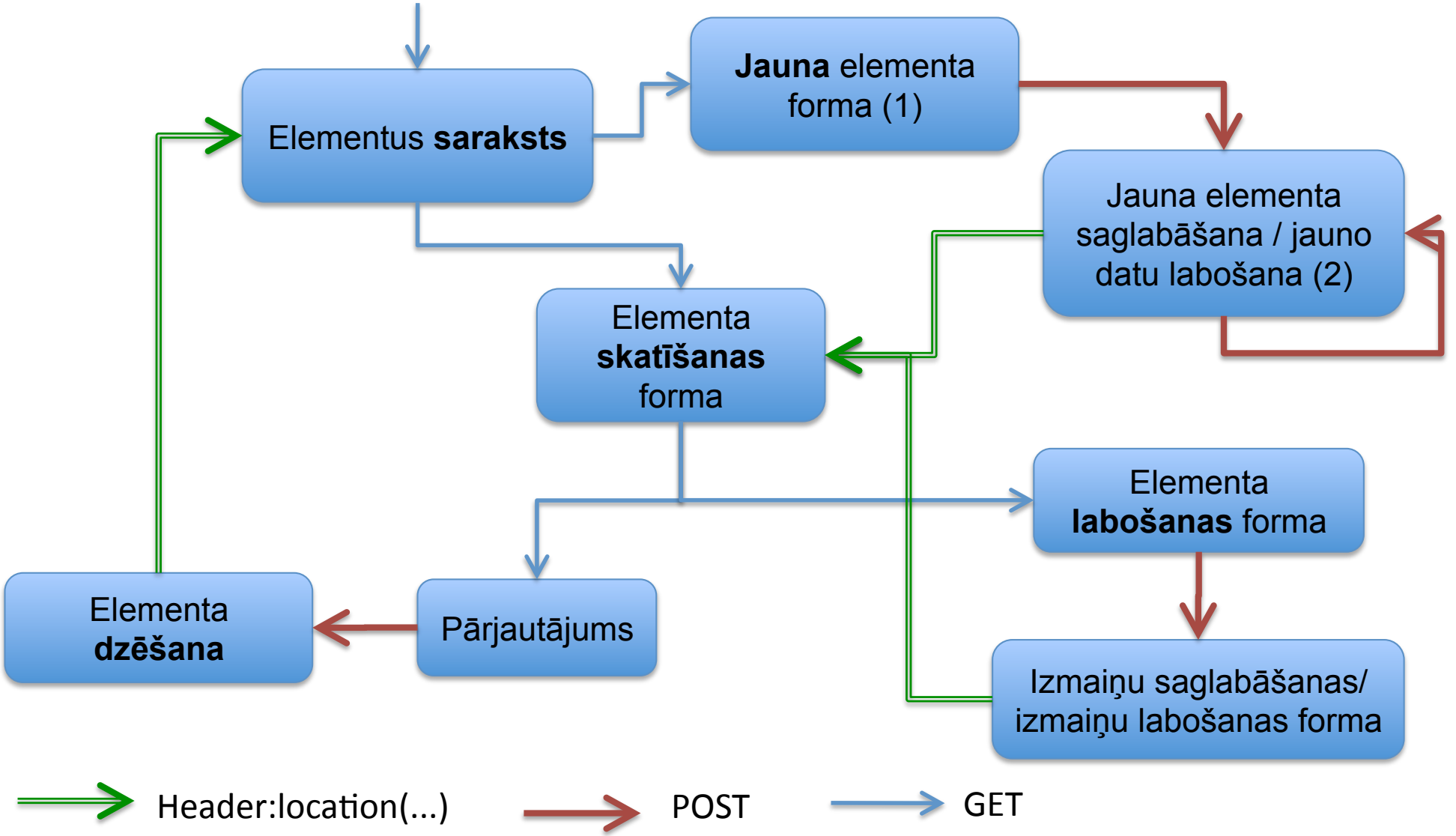
Elementa dzēšanas funkcionalitāte

- Vēkams lietotājam pārjautāt, vai tiešām dzēst ierakstu
- Pārjautāšana ir atsevišķa lapa, kurai var padot GET parametru (vēl nekas netiek mainīts datu glabātuvē)
- Apstiprinājumu sūta caur POST

Dzēšot veicamās darbības

- Lietotāja identifikācija un autorizācija
  - vai lietotājs drīkst dzēst šo ierakstu
- Datu integritāte
  - Vai ieraksta trūkums neizraisa problēmas citās datu glabātuvēs?
- Ja nepieciešama "undelete" funkcionalitāte, ierakstus nevar dzēst datu glabātuvē, tikai jāuzstāda "dzēsts" statuss!

CRUD: kopaina



KODA UZLABOŠANAS IESPĒJAS

DRY

- **D**on't **R**epeat **Y**ourself!
- Koda kopēšana no vienas risinājuma daļas uz citu liecina par problēmu kodā.
- Risinājumi
  - nodalīt vairākkārt izmantojamo funkcionalitāti funkciju bibliotēkā
  - pārveidot kodu par objektorientētu kodu
  - veidot universālākus koda fragmentus (piem., *new.php* un *new_submit.php* var apvienot)

Zināšanu koncentrācija

- *Every piece of knowledge must have a single, unambiguous, authoritative representation within a system.*
- Uz vienu un to pašu jēdzienu attiecināmais kods jāglabā vienkopus
- Labi risināms ar OO un moduļu bibliotēkām

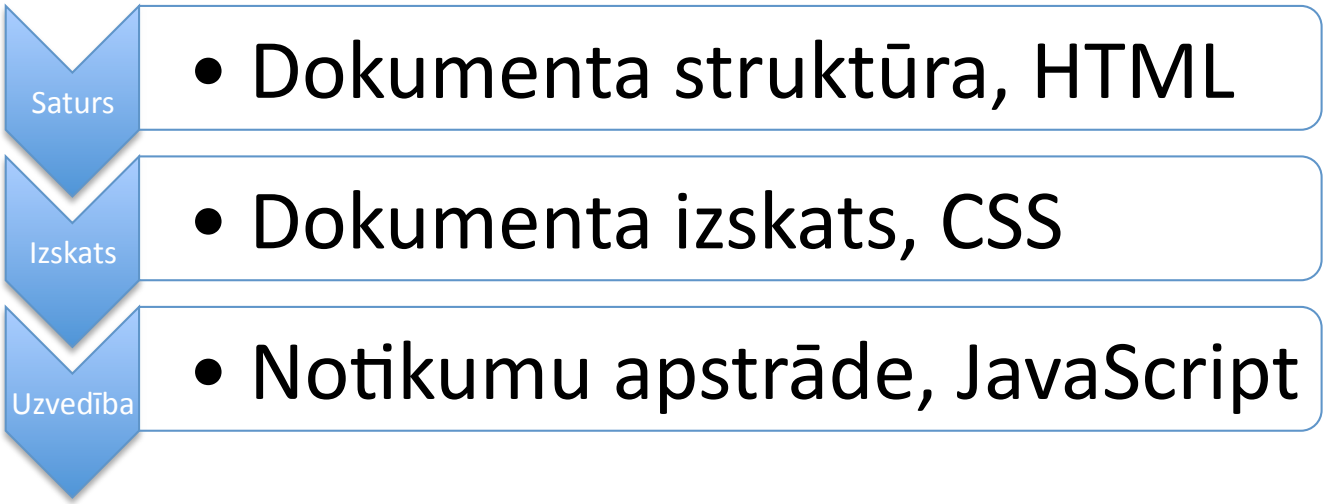
Programmatūras slāņu nodalīšana

- Datu līmenis, biznesa loģika, prezentācijas līmenis
- Var nodalīt:
  - kods, kas darbojas ar DB, saņemtajiem datiem, lietotāju autentifikāciju (biznesa loģika)
  - kods, kas ģenerē HTML formas

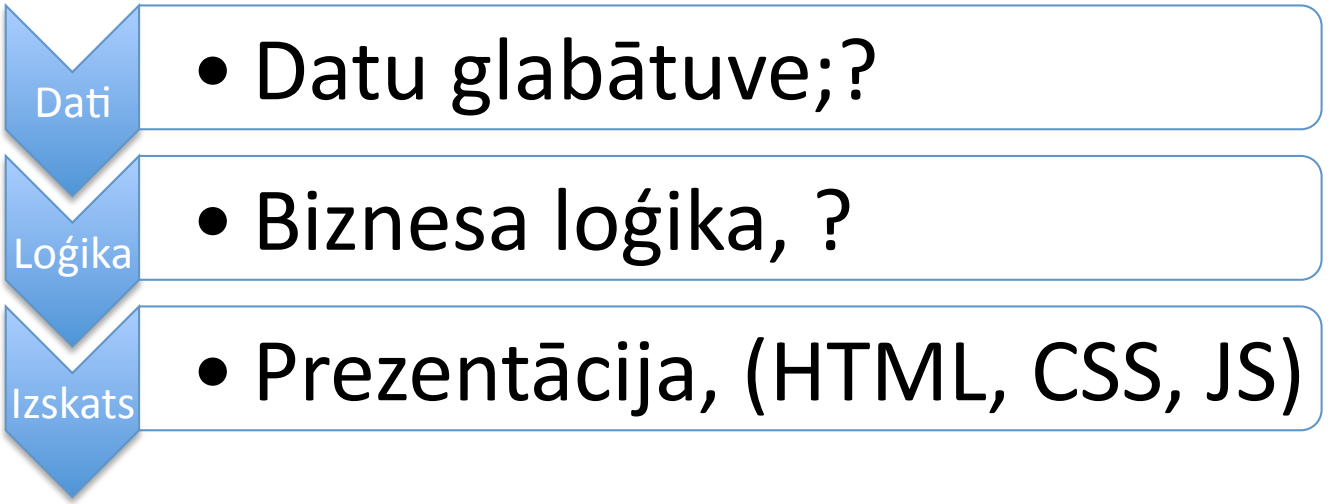
PROGRAMMATŪRAS SLĀŅOŠANA

Funkcionalitātes noslāņošana tīmekļa programmatūrā

Funkcionālie slāņi klienta puses tehnoloģijās



Funkcionālie slāņi servera puses tehnoloģijās



Datu līmenis

- Datu glabātuve, kas "neko nezina" par tās lietojumu
- Nodrošina (relatīvi) drošu datu glabāšanu
- Tipiski - relāciju datu bāze (bet var arī būt failu sistēma, POP3/IMAP, Web servisi)

- Lai strikti nodalītu datu līmeni no biznesa līmeņa, mēdz izmantot "stored procedures" un "triggers" (SQL gadījumā)
- Tādējādi datu līmeni un biznesa līmeni var realizēt dažādi izstrādātāji, kas vienojušies par interfeisu

Biznesa modelis

- Definē sistēmā esošos biznesa jēdzienus un to savstarpējās attiecības
- Pārvalda datu līmenī esošo informāciju
- “Neprot” veikt prezentācijas līmeņa funkcijas, bet nodrošina visus tam nepieciešamos servisus
- Piemēram, "Phone" klase
  - spēj nolasīt datus no "phone" tabulas
  - spēj saglabāt savus datus atpakaļ tabulā
  - māk noteikt, kurā valstī tālruņa numurs reģistrēts

Prezentācijas līmenis

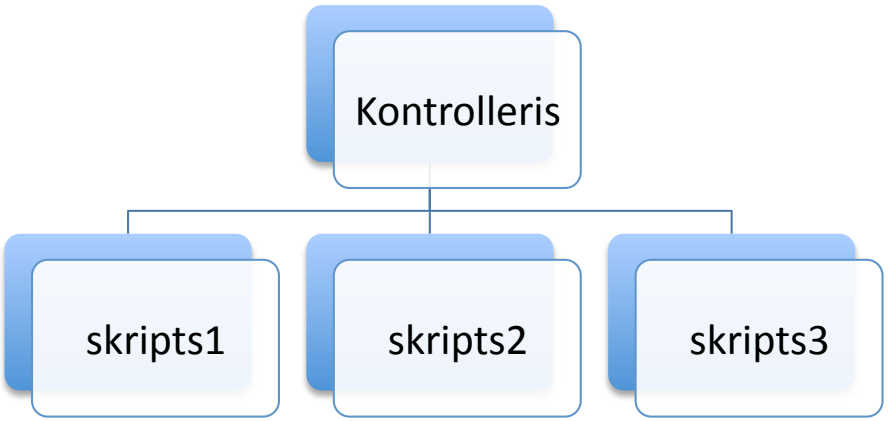
- Prezentācijas līmenis ir sistēmas seja, kādu to pazīst lietotājs
  - Uzraksti, attēli, pogas datu ievadīšanas lauki
  - Kārtošana, meklēšana
  - Navigējamība
- Prezentācijas līmenis sadarbojas ar biznesa modeļa līmeni. Tas "negrābstās" gar datu līmeņa servisiem.

Trīsrindu arhitektūra: piemērs

- Datu bāzē:
  - tabulas "employee", "phone"
  - SQL procedūra getEmployeePhones
- Biznesa modelī kases
  - Employee, EmployeetList
  - Phone, PhoneList
- Prezentācijas līmenī
  - Darbinieku saraksts ar saiti uz detalizētas info skatīšanu
    - izmanto EmployeeList un Employee klases
  - Darbinieka skatīšanas lapā – darbinieka tālruņu saraksts
    - izmanto Employee klasi, PhoneList klases

CENTRĀLĀ KONTROLLERA PIEEJA

Centrālais kontrolleris



- Kontrolleris apstrādā visus ienākošos pieprasījumus
- Atkarībā no ieejas datiem un pašreizējā stāvokļa, piemeklē skriptus, ko varētu izpildīt

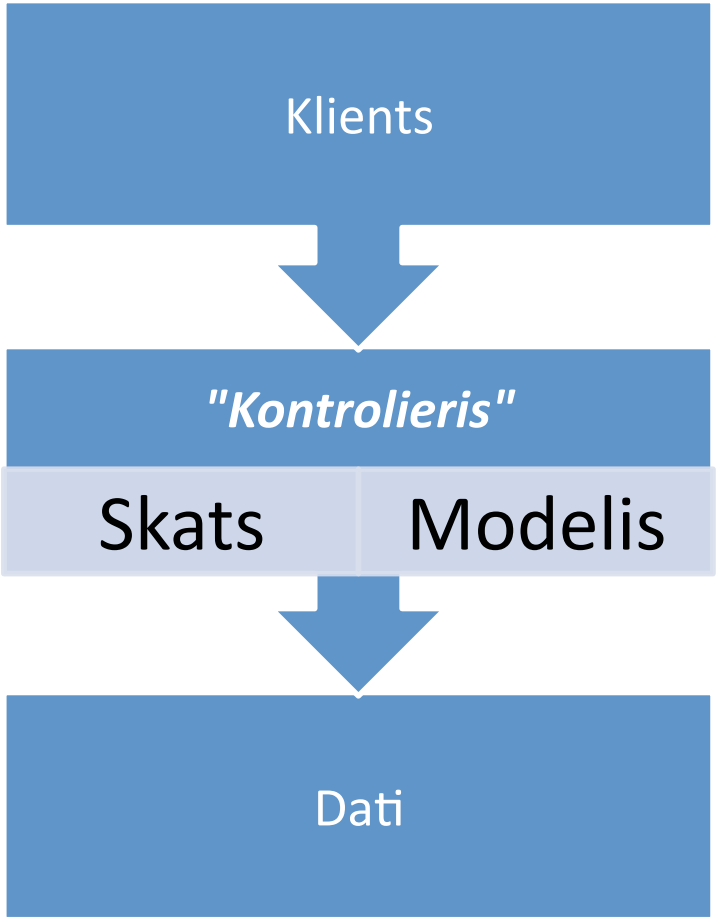
Ieguvumi no centrālā kontrollera

- Iespēja centralizēt vairākkārt izmantotu kodu
  - parametru pārbaudes
  - autentifikācija
- Iespēja kontrolēt lietotnes izpildes gaitu
- Var ātri mainīt veidu, kā apstrādā VISUS pieprasījumus

MVC PRINCIPS

Kas ir MVC

- MVC = **M**odel, **V**iew, **C**ontroller
- Trīsrindu arhitektūras papildinājums, kas noderīgs tīmekļa lietotnēs
- Precizē lietotnes prezentācijas un navigējamības loģiku



Kontrolieris MVC arhitektūrā

- Kontrolieris
  - Saņem ienākošos pieprasījumus
  - Saņem URL vaicājuma (*query*) un formu parametrus
  - izsecina, kura darbība veicama
  - Izmanto modeli un skatus, lai veidotu atbildi
- Var veikt papilddarbus:
  - ievaddatu pārbaude, datu filtrēšana
  - piekļuves tiesību pārbaudes
  - izņēmuma gadījumu, t.sk. neeksistējošu adrešu apstrāde

Modelis MVC arhitektūrā

- Modelis “atbildīgs” par pareizu darbu ar datiem
- Bieži vien – paslēpj SQL lietošanu no pārējām komponentēm

Skats MVC arhitektūrā

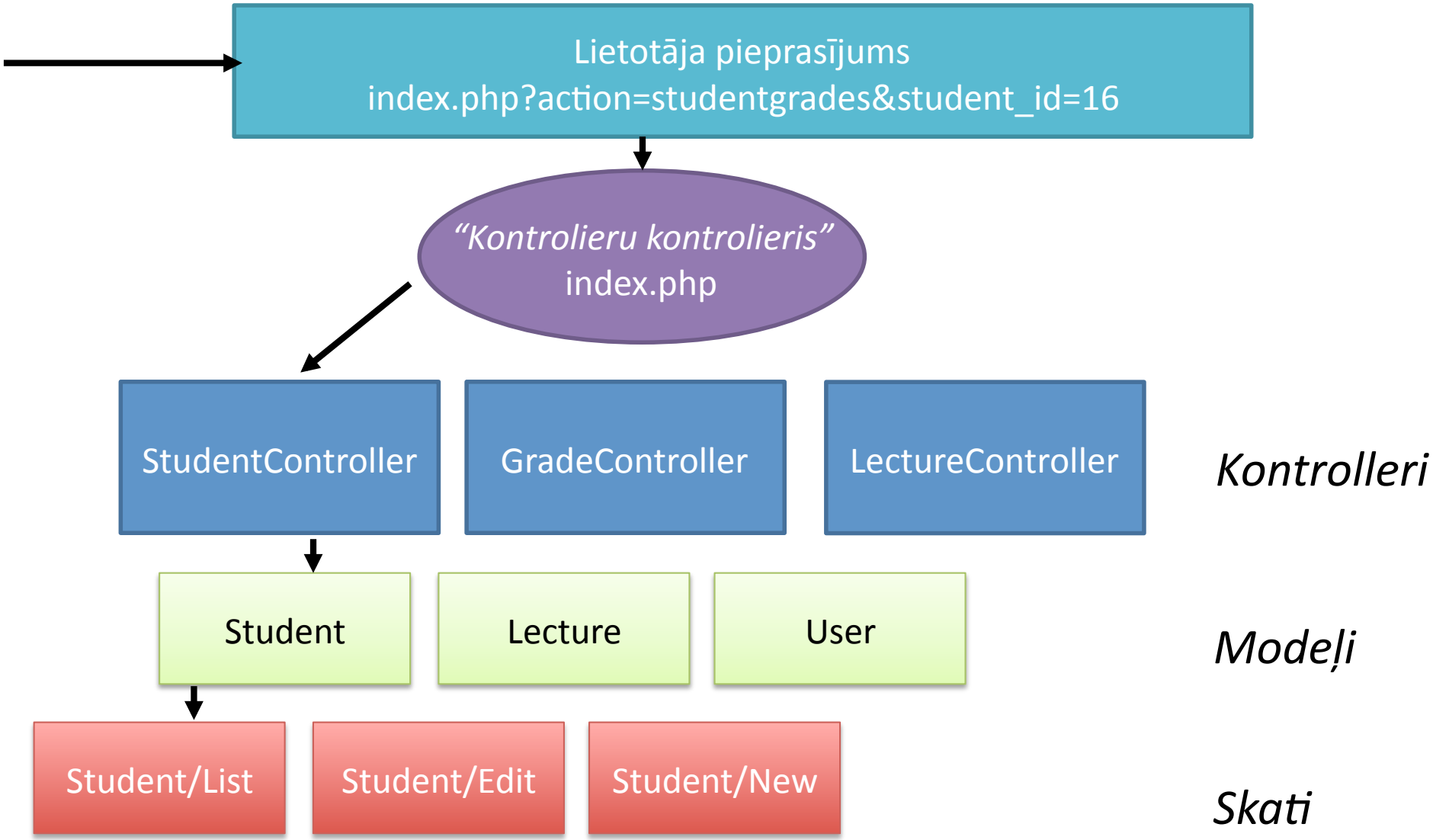
- Skats realizē *tīrās* prezentācijas līmeņa funkcijas
- Analoģisks prezentācijas līmenim vienkāršajā trīsrindu arhitektūrā, bet atbrīvots no "blakus darbu" izpildes
- Mēdz izmantot šablonu valodas, lai nodalītu arī valodas

PHP "rāmji" ar MVC atbalstu

- MVC ir ideoloģija, to var realizēt katrs pats; galvenais ir domāšanas veids, nevis konkrētā realizācija

- Ir vairāki rāmji (*framework*), kas piedāvā standartizētus MVC risinājumus.
  - http://www.codeigniter.com
  - http://www.cakephp.org/
  - http://framework.zend.com/
  - http://www.phpmvc.net/
  - http://www.symfony-project.org/
- Rāmji *uzspiež* savu ideoloģiju (piemēram, savus kontrolierus jāglabā katalogā /controllers, savus skatus jāglabā katalogā /views), bet tas ir **audzinoši**.

MVC – vizuāli


Lietotāja pieprasījums
index.php?action=studentgrades&student_id=16
"Kontrolieru kontrolieris"
index.php
StudentController
GradeController
LectureController
Kontrolleri
Student
Lecture
User
Modeļi
Student/List
Student/Edit
Student/New
Skati

Uzdevums mājās

- Noskaidrot, kas ir kas
  - Front Controller Pattern
    - http://www.phpwact.org/pattern/front_controller
    - http://www.phppatterns.com/docs/design/the_front_controller_and_php
  - Model-View-Controller
    - http://www.phpwact.org/pattern/model_view_controller

CRUD + skaistās adreses = REST

- REST
  - Representational State Transfer
  - Autors: Roy Fielding, 2000. gads
- Ideja:
  - URL identificē resursa veidu un pašu resursu, tas paliek nemainīgs
    - http://localhost/users/145/
  - Lieto dažādas HTTP komandas
    - GET – lasa informāciju par resursu
    - POST – veido jaunu resursu
    - PUT- labo esošu resursu
    - DELETE - dzēš resursu
- Problēma: pārlūkprogrammas pašlaik neietver vairāk kā GET un POST