VILNIAUS UNIVERSITETAS

MATEMATIKOS IR INFORMATIKOS FAKULTETAS
INFORMATIKOS KATEDRA

PRAKTIKOS ATASKAITA

Praktiką atliko: **Laimonas Beniušis**
(studento vardas, pavardė) (parašas)

Informatika, Kompiuterių Mokslas, 4 kursas
(studijų programa, pakopa, kursas)

Praktikos institucija: Lietuvos kriminalinės policijos biuras
(organizacijos pavadinimas)

Organizacijos praktikos vadovas: ITS viršininkas, Jonas Gurskas
(pareigos, vardas, pavardė)

Organizacijos praktikos vadovo įvertinimas: _____
(įvertinimas, parašas)

Universiteto praktikos vadovas: _____
(mokslo laipsnis, vardas, pavardė)

_____
(parašas)

Ataskaitos įteikimo data _____
Registracijos Nr. _____
Įvertinimas _____
(data, įvertinimas, parašas)

Vilnius, 2018

Turinys

1. Praktikos vietos aprašymas

1.1 LKPB

Lietuvos kriminalinės policijos biuras – centrinė kriminalinės policijos įstaiga, veikianti visoje šalies teritorijoje. LKPB veikla apima:

- Visų dešimties apskričių vyriausiųjų policijos komisariatų veiklos kontroliavimas ir koordinavimas
- Kriminalinės policijos veiklos strategijos formavimas
- Lietuvos policijos įstaigų ir kitų teisėsaugos institucijų tarptautinio bendravimo su Interpolu, Europolu, SIRENE ir kitų šalių teisėsaugos institucijų vykdymas ir koordinavimas
- Kriminalinės žvalgybos mokymų organizavimas

LKPB užkardo, atskleidžia bei tiria:

- Sunkius ir labai sunkius tarpregioninio ar tarptautinio pobūdžio nusikaltimus
- Nusikalstamas veiklas, darančias didelę žalą valstybei ar asmenims
- Labai aukšto lygio organizuotų nusikalstamų grupių daromus nusikaltimus
- Teroro aktus
- Nusikaltimus nuosavybei, turtinėms teisėms ir turtiniams interesams, ekonomikai ir verslo tvarkai, finansų sistemai
- Nusikaltimus elektroninėje erdvėje
- Sunkius ir labai sunkius nusikaltimus, susijusius su neteisėta narkotikų apyvarta
- Kitas nusikalstamas veiklas

1.2 Darbo sąlygos

Lietuvos kriminalinės policijos biuras yra įsikūręs Saltoniškių g. 19, Vilniuje. Jis yra suskirstytas į vadybas, kurios dar yra skirstomos į skyrius. Informacinių technologijų skyrius (ITS) yra 11-ame aukšte, kuris priklauso informacinių technologijų vadybai (ITV). Darbuotojai suskirstyti kabinetuose po vieną arba dviese. Darbo vietos priemonės skiriasi nuo darbo pobūdžio. Man buvo skirtas atskiras darbo kabinetas turintis:

- Darbo stalą
- Stacionarus kompiuterį
- Du 27 colių monitorius
- Nenutrūkstamo maitinimo šaltinį
- Klaviatūrą, pelę

Programinė įranga darbui nėra apribojama, t. y. galima pasirinkti savo nuožiūra. Dažniausiai užduotys atliekamos individualiai (keli asmenys to paties projekto kodo nerašo), tačiau būna išimčių.

2. Įvadas

LKPB yra atsakingas už teisėsaugos procesus, kuriems dažnai yra naudojamos informacinės sistemos. Informacinių technologijų skyrius palaiko, prižiūri, plečia bei kuria tokias sistemas.

Man buvo paskirta individuali užduotis sukurti įgyvendinti programą, kuri būtų skirta įvairaus pobūdžio psichologiniams testams vykdyti, bei rezultatams fiksuoti. Programa bus naudojama LKPB darbuotojų tyrimui.

Praktikos tema: Kompiuterizuotų pažintinių užduočių rinkinys
Uždaviniai:

1. Susipažinti su nauju Java WEB karkasu (angl. *Framework*) bei gilinti esamas žinias
2. Pasirinkti HTML5 *Canvas* JavaScript biblioteką, kuri tenkina užduoties reikalavimus
3. Gilinti HTML, JavaScript ir Java žinias
4. Įgyvendinti programą (suprojektuoti ir suprogramuoti) su pasirinktu Java WEB karkasu pagal gautą aprašymą.

Praktikos vadovas paskatino gilinti žinias ir specializuotis Java srityje, nes ITS skyriuje nėra darbuotojų, su giliomis Java žiniomis. Buvo pateiktas užduoties techninis aprašas, kuris buvo pagrindinis informacijos šaltinis, kaip turi veikti testai. Taip pat vyko komunikacija su psichologu, kuris yra šio užduočių rinkinio užsakovas. Nors ir užsakovas siūlė daryti programą, kuri veiktų lokaliai, tačiau, dėl palaikymo ir atnaujinimo paprastumo, praktikos vadovas nutarė daryti WEB pagrindu. Taip pat WEB srityje gilinti žinias yra ypač aktualu dabartinėje IT srityje, kai dauguma informacinių sistemų pasitelkia internetą informacijos perdavimui.

3. Teorija ir informacijos šaltiniai

Užduoties atlikimui buvo aktualios objektinio programavimo, duomenų bazių valdymo sistemų bei kompiuterinės grafikos žinios, įgytos universitete. Didžioji užduoties dalis buvo HTML5 *Canvas* valdymas per pasirinktą JavaScript biblioteką.

3.1 Paper.js

Paper.js [P18] yra JavaScript biblioteka, skirta vektorių grafikos valdymui, kurios pagrindas yra HTML5 *Canvas*. Ši biblioteka turi daugybę funkcijų ir galimybių:

- Lengvas figūrų valdymas *Canvas Path* objekto pagrindu
  - Daug jau paruoštų figūrų (apskritimas, elipsė, stačiakampis, žvaigždė, reguliarus poligonas)
  - Figūros formavimas tiesiog pridedant tašką į pasirinktą *Path* objektą
- Interaktyvumas
  - Lengvas objektų valdymas pagal vartotoju veiksmus su pelyte ar klaviatūra
- Dokumentų objektų modelis
  - Naudoja griežtą objektų hierarchiją *PaperScope* → *Project* → *Layer* → *Item*
  - Kiekvieną *Canvas* modifikacija yra atskiras objektas, todėl taip išlaikoma tvarkinga sistema, kurią lengva valdyti bei plėsti
- Paprastas teksto formatavimas su atributais
- Pokyčiai matomi vos pakeitus aktyvaus matomo sluoksnio objekto atributą

3.2 Ninja framework

Pasirinktas Java WEB karkasas yra *Ninja framework* [N18], kuris turi šias funkcijas:

- REST tipo architektūra.
- Puslapių generavimas
- Vartotojų sesijų ir informacijos valdymas
- Efektyvus JSON tipo informacijos perdavimas
- Duomenų bazės valdymas
- Greitas aplikacijos kūrimas bei pokyčių matymas naudojant specialų rėžimą, kuris skenuoja projekto direktoriją ir, aptikus pokyčius, perleidžia aplikaciją

Ninja framework pats visko nedaro, o pasitelkia atviro kodo bibliotekas didžiajai daliai funkcijų atlikti:

- *Google Guice* biblioteką, priklausomybių injekcijų valdymui (angl. *Dependency injection*)
- ORM (angl. *Object Relational Mapping*) tipo sistemą per standartinį JPA [JCE+03] (*Java Persistence API)* su *Hibernate* varikliu
- *Apache Freemarker* internetinių puslapių šablonų generavimo sistemą (angl. *template engine)*
- *i18n* teksto valdymo ir internacionalizavimo mechanizmą, kuris susietas su *Apache Freemarker* sistema
- *WebJars* JavaScript bibliotekų valdymo sistemą
- *Maven* Java bibliotekų valdymo sistemą

3.3 Kompiuterizuoto pažintinių užduočių rinkinys

Rengiamas kompiuterizuotas pažintinių užduočių rinkinys. Tai yra priemonė įvertinti asmenų pažintinius gebėjimus. Šią metodiką sudaro aštuonios pažinties funkcijas tiriančios užduotys. Rinkinys sukurtas remiantis V. Jurkuvėno ir bendraautorių [JBG06] darbais ir PEBL užduočių rinkinio (angl. *PEBL test battery*) .

Užduočių rinkinys sudarytas iš:

- Psichomotorinės:
  - Mygtuko spaudymo testas (angl. *Finger tapping*). Klasikinė paprastą motorinį gebėjimą matuojanti užduotis: Dominuojančia ir nedominuojančia ranka paspausti mygtuką kaip galima greičiau.
- Dėmesingumo
  - Reakcijos laiko testas (angl. *Simple reaction time*). Atsako greitį matuojanti užduotis: paspausti mygtuką kuo greičiau signalo ekrane pasirodymo.
- Atminties
  - Skaičių serijos atsiminimo testas (angl. *Immediate serial recall*). Trumpalaikės skaitinės atminties apimtį matuojanti užduotis: Stebėti ekrane pasirodžiusius skaičius, ir matytą seką pakartoti.
  - Skaičių serijos atgalinio atsiminimo testas. Analogiškai, tik skaičių seką suvesti nuo galo.
  - Corsi kubelių testas (angl. *Corsi block test*). Trumpalaikės vaizdinės atminties apimtį matuojanti užduotis: Stebėti užsidegančių kubelių seką ir ja pakartoti.
- Erdvinio suvokimo
  - Objektų atpažinimo užduotis (angl. *Object judgement*). Sudėtingos informacijos apdorojimo greitį matuojanti užduotis: Stebėti ekrane pasirodžiusią figūrą, ir įvertinti naują pasirodžiusią figūrą, ar tokia pat, tik kitu kampu, ar jau kitokia. Figūros generuojamos pagal F. Attneave pristatytą algoritmą [AA56].
- Vykdomosios
  - Modifikuota Viskonsino kortelių atrankos užduotis (angl. *Berg Wisconsin Card Sorting Test)*. Vykdomosios taisyklių supratimą ir keitimą matuojanti užduotis: Rodoma kortelė, su skiriamaisiais bruožais (spalva, simbolių forma, simbolių kiekis). Reikia priskirti kortelę vienai iš 4 grupių, pagal nežinomą taisyklę. Tik gaunant atsaką „Teisingai“ arba „Neteisingai“ reikia suprasti taisyklę.

- Stroop užduotis (angl. *Stroop test*). Vykdomąją funkciją, pažintinį lankstumą, atrankinį dėmesį, kognityvini slopinimą bei informacijos apdorojimo greitį matuojanti užduotis: Pasirodo ekrane spalvoti žodžiai, tiriamajam reikia teisingai nurodyti pasirodžiusio žodžio spalvą. Žodžiai gali sutapti arba nesutapti su spalva, arba gali būti visai nesusiję su spalvomis.

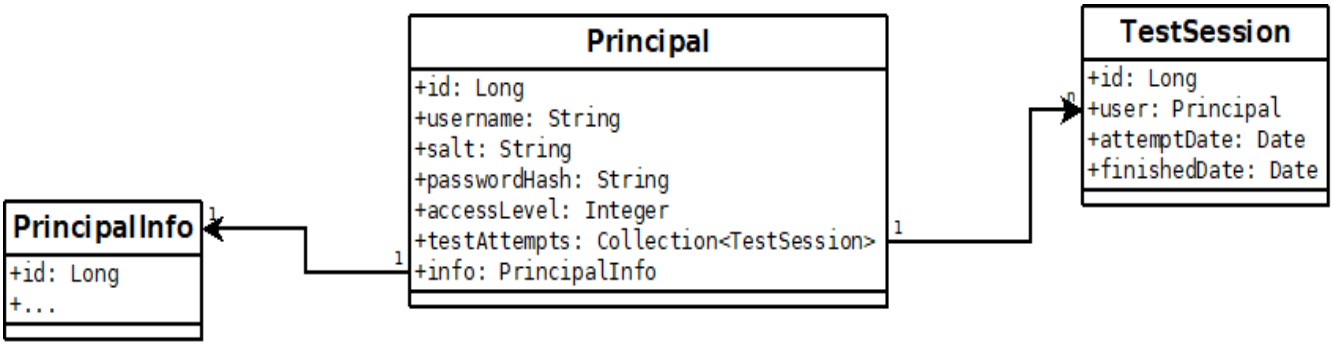
4. Įgyvendinimas

4.1 Duomenų bazės architektūra

Šiai aplikacijai įgyvendinti, reikėjo duomenų bazės struktūros, kuri galėtų efektyviai išsaugoti bei skaityti išsaugotus duomenis. Kadangi yra naudojamas ORM įrankis, duomenų bazė gali būti lengvai generuojama iš esančių Java klasių. Kūrimo metu, buvo naudojama *H2* lokali duomenų bazė, dėl jos greičio ir naudojamo paprastumo, tačiau realioje aplinkoje bus naudojama *Microsoft SQL Server* arba *Oracle Database* (dar nenuspręsta). Kadangi ORM pašalina pačių SQL komandų rašymą, prijungti prie tuščios duomenų bazės įgalina kelių eilučių konfigūracijos faile pakeitimas.

4.1.1 Vartotojų modelis

Kadangi aplikacija naudoja REST architektūros, reikia sistemos, kuri leistų sekti vartotojo stadiją, prieigos teisių lygį, bei rezultatus.

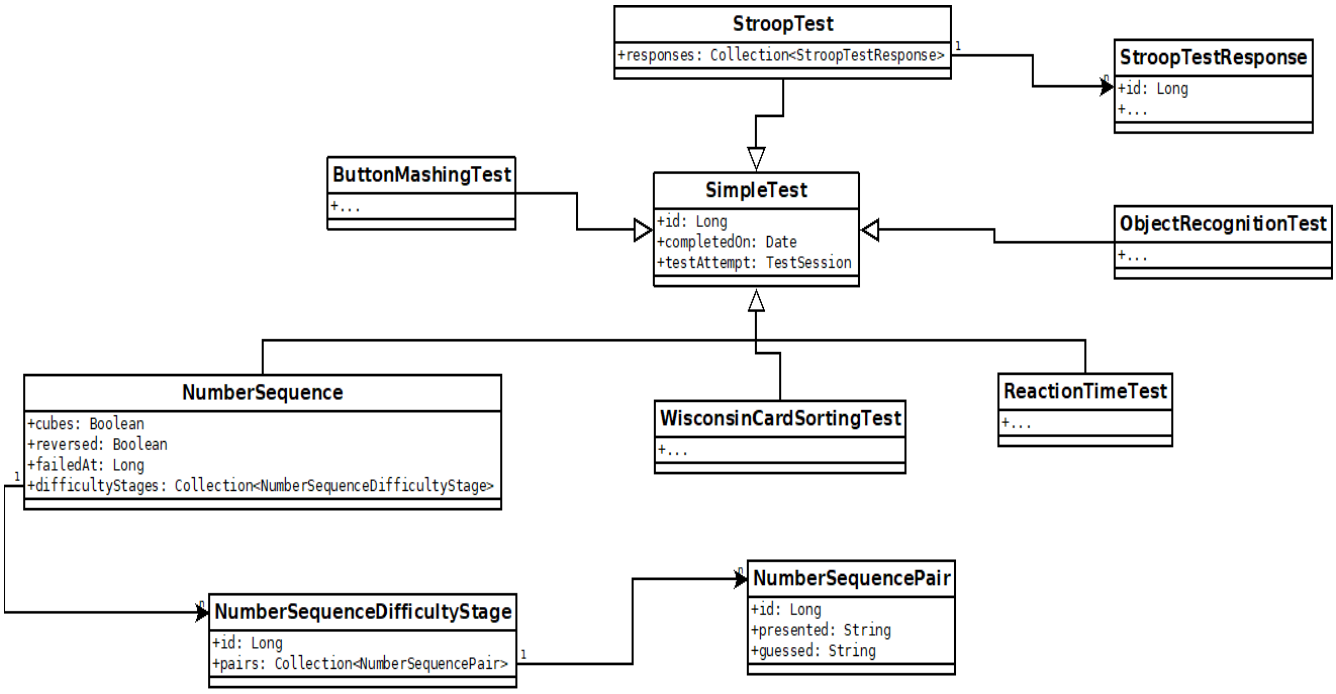


1 Pav. Vartotojų modelių klasių diagrama.

Minimali vartotojo informacija yra aprašoma *Principal* klasėje. Kadangi dalis būsimų testo dalyvių negali atskleisti savo tapatybės, tiesiog registruojasi su išduotu kodu ir *PrincipalInfo* klasė nėra naudojama. Kiekvienas vartotojas gali turėti kelias testo sesijas, aprašytas *TestSession* klase, kuri vėliau yra susiejama su bazine testo klase. Numatyta 3 lygių vartotojų teisių sistema: Administratorius, Personalas ir Vartotojas. Vartotojas gali tik prisijungti ir atlikti testus numatyta tvarka. Personalas gali atlikti testus bet kokia tvarka bei peržiūrėti ir eksportuoti informaciją. Administratorius gali daryti viską, ką ir Personalas bei yra naudojamas keisti vartotojų teises, nes Personalo skaičius iš anksto nėra žinomas.

4.1.2 Testų modelis

Pagal techninį aprašą, yra 8 skirtingos užduotys (testai). Informacijai išsaugoti buvo sukurtas duomenų klasių modelis, naudojantis paveldėjimą, ką leidžia ORM įrankis.



2 Pav. Testų rezultatų klasių diagrama.

Kiekvienas iš 8 skirtingų testų paveldi bazinę klasę *SimpleTest*, kuri turi sąsają su klase *TestSession*, kuris yra diferencijuojantis atributas leidžiantis atskirti kuriai vartotojui sesijai priklauso specifinis testas. Visi atminties kategorijos testai dalinasi tą pačia modelio medžio šaka prasidedančia *NumberSequence*. Paveldėjimas yra *InheritanceType.JOINED* tipo, todėl nėra daugybės tuščių laukų. Gali būti lėtesnis informacijos ieškojimas, tačiau supaprastina jos valdymą ir eliminuoja kodo dubliavimą. Kitų lentelinių klasių atributai yra paslėpi

4.2 Serverinė aplikacijos dalis

Kadangi *Ninja Framework* naudoja DI (angl. *Dependency Injection*) mechanizmą, rekomenduojama, kad visos klasės, būtų abstrakčios arba interfeisai, ir jų implementacija būtų kitur. Tokiu būdu atskiriamos konkrečios klasės nuo bendros aplikacijos logikos ir gali būti lengvai keičiamos klasių implementacijos, nepakeičiant naudojamo interfeiso.

Sukurti teisingai GET ir POST kelių sąsajos, su tam tikro valdiklio metodu. Kur reikia didesnio teisių lygio, sukurti filtrai. Kiekvieno testo pabaigoje, yra keičiamas vartotojo sesijos parametras, norint atsekti kurį testą dabar vartotojas atlieka. Taip pat testų informacija yra perduodama JSON formatu (išskyrus apmokomoje dalyje), kuris yra formuojamos į atitinkamo testo klasės objektą ir išsaugojamas duomenų bazėje.

4.3 Klientinė aplikacijos dalis

Sukurti 2 *Apache Freemarker* maketai (angl. *layout*): testinis ir pagrindinis. Maketai yra patogūs tuo, kad nereikia kopijuoti viso kodo, jeigu kažkuri puslapio dalis pasikartoja (pvz. Antraštė, arba poraštė) tereikia įterpti pasikeitusią pagrindinę dalį. Taip pat *Apache Freemarker* leidžia aprašyti logiką, kuri bus vykdoma puslapio generavimo metu, pvz.: nerodyti tam tikro mygtuko, jeigu vartotojas neturi numatytos teisės. Galima perduoti parametrus ar duomenų bazės užklausos informaciją iš aplikacijos (pvz.: lentelės HTML kodui generuoti), kas leidžia sumažinti kreipinių skaičių, bei kodo dubliavimą.

Aplikacijos pagrindinė dalis apipavidalinta *Bootstrap* stiliumi, kuri apima rezultatų peržiūrą, vartotojų prisijungimą, registraciją, bei teisių valdymą.

4.3.1 *Paper.js* problemos ir sprendimai

Aplikacijos testinė dalis yra vienodo pagrindo – HTML5 *Canvas* užimanti visą ekraną, kuri pasitelkia *Paper.js* biblioteką vaizdų kūrimui. Visa aplikacijos logika ir veiksmų seka rašoma ES6 standarto JavaScript.

Kadangi visos užduotys turi stadijas, kurios keičiasi priklausomai nuo tiriamojo veiksmų, reikia kažkokios skirtingų stadijų sistemos, kuris jas kontroliuotų ir galėtų atkurti, kai to reikia. *Paper.js* tokio mechanizmo neturi, nes visi pokyčiai išlieka, todėl norint gauti pradinį vaizdą reikia viską perkurti arba gaminti vaizdų kopijas. Taip pat, *Paper.js* naudoja absoliučias koordinates, todėl

tokiu būdu kurtas vaizdas priklauso nuo monitoriaus dydžio ir rezoliucijos. Šioms problemoms išspręsti buvo sukurtas globalus objektas *PaperContext*, kuris kontroliuoja skirtingus *PaperScope* (pradinis hierarchijos modelio objektas).

Stadijos atkūrimo problema buvo išspręsta visų elementų kūrimą sudedant į anonimines funkcijas ir jas sugrupuojant su atitinkamu *PaperScope* objektu. Tokiu būdu sudaromas pakrovos funkcijų sąrašas, kuris perleidžiamas kiekvieną kartą perkuriant stadiją. Visi objektai, sukuriami pakrovos funkcijose, yra automatiškai įtraukiami į aktyvų *PaperScope* objektą.

Skirtingų rezoliucijų problemai išspręsti buvo įvesta *RelativePoint* klasė, kuri naudoja reliatyvias koordinates vietoj absoliučių. Prieš piešiamojo objekto sukūrimą, yra proporcingai apskaičiuojamos ir jam priskiriamos absoliučios koordinatės pagal esamo vartotojo ekrano dydžio matmenis. Taip pat, prieš užkraunant naują stadiją, visi sukurti objektai yra pamažinami (arba padidinami) pagal mastelį, kuris gaunamas iš esamo dydžio ir bazinio (kuris buvo naudojamas aplikacijos kūrimui). Tokiu būdu yra išsprendžiama vaizdų suvienodinimo skirtingose rezoliucijose problema.

Skirtingose stadijose reikalingi skirtingi įvykių valdikliai (angl. *Event handler*). Tradiciškai, vartotojo sukelti įvykiai (klaviatūros ar pelytės veiksmai) yra globalūs. Sukurta įvykių valdymui skirta klasė, kuri buvo panaudota globaliems įvykiams valdyti, bei integruota į *PaperContext*, savo ruožtu lokalizuojanti įvykius (nukreipia į aktyvaus *PaperScope* objektą, kuris turi po lokalų įvykių valdiklį). Ši įvykių sistema, palaiko 3 tipų įvykius:

- *instant* – veiksmas atliekamas nedelsiant
- *delayed* – veiksmas atliekamas palaukiant nurodytą laiko tarpą
- *delayedUnique* – veiksmas atliekamas palaukiant nurodytą laiko tarpą, bet pakartotini jo iškvietimai yra blokuojami, kol nepasibaigė inicijuojantis laukimas. Taip pat šiuos veiksmus galima susieti, kad iškviestas veiksmas būtų blokuojamas, jeigu yra nebaigtų su juo susietų veiksmų.

Naudojant tokias sistemas, pačios aplikacijos logikos įgyvendinimas buvo gana paprastas.

5. Išvados

5.1 Bendros

Mokantis naują karkasą ar biblioteką, verta susipažinti su didžiąja jos funkcionalumų ir galimybių dalimi, prieš pradedant spręsti konkrečią užduotį. Tokia metodika padeda užtikrinti, kad pasirinkta užduoties sprendimo strategija yra tinkama ir nereikės perrašyti dalies kodo, atradus anksčiau nematytą funkcionalumą ar išsiaiškinus, kad pasirinkta strategija netinkama, aptikus nenumatytą kliūtį.

5.2 Pasiekti rezultatai

Atliekant praktiką LKPB buvo:

- Pakelta asmeninė kvalifikacija:
  - Gilintos Java žinios JPA srityje bei JavaScript žinios
  - Išmoktas ir pritaikytas *Ninja Framework* karkasas
  - Išmokta ir panaudota JavaScript *Paper.js* biblioteka
  - Įgyta darbo praktika
- Pasiekti institucijai aktualūs rezultatai:
  - Įgyvendinta WEB aplikacija pagal jos aprašą

5.3 Praktikos privalumai ir trūkumai

Atliekant praktiką LKPB buvo privalumų bei trūkumų:

Privalumai:

- Atliekamas darbas turi tiesioginę taikomąją vertę
- Universitete įgytos teorinės žinios buvo pritaikomos praktikoje
- Galima buvo dirbti ties viena užduotimi, kas retai pasitaiko
- Kolegų idėjos padėjo priimti teisingus dizaino sprendimus
- Įgyta darbo koordinavimo bei derinimo kartu su kitos srities atstovu praktika

Trūkumai:

- Nebuvo komandinio darbo praktikos
- Užduoties pagrindinis funkcionalumas buvo gana paprastas

Literatūra

[Apa18] Apache Freemarker. (2018). https://freemarker.apache.org/

[AA56] Attneave, F., & Arnoult, M. D. (1956). The quantitative study of shape and pattern perception. *Psychological Bulletin*, 53(6), 452-471.

[JBG06] Jurkuvėnas V., Bagdonas A., Germanavičius A. (2016). Simple and complex information processing speed in psychiatric samples. Vilniaus Universitetas

[JCE+03] Jendrock E., Cervera-Navarro R., Evans I., Gollapudi D., Haase K., Markito W., Srivathsa C. (2013). The Java EE6 Tutorial. p. 579-601. https://docs.oracle.com/javaee/6/tutorial/doc/javaeetutorial6.pdf

[N18] Ninja framework. (2018). http://www.ninjaframework.org/

[P18] Paper.js (2018). http://paperjs.org