VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS

Profesinės darbo praktikos įmonėje EPAM sistemos ataskaita

Report on Professional Work Practice in the Company EPAM systems

Profesinės praktikos ataskaita

Atliko:	2 kurso studentas Matas Savickis	(parašas)
Universiteto praktikos vadovas:	Petrauskas Karolis, Doc., Dr.	(parašas)
Įmonės praktikos vadovas:	Nerijus Tenys	(parašas)

Vilnius – 2021

TURINYS

1. Įvadas

1.1. Praktikos vienos pasirinkimo motyvacija

Pasirinkau atlikti praktiką įmonėje EPAM sistemos, nes norėjau įgauti patirties dirbant tarptautinėje korporacijoje dirbant su trumpalaikiais klientų užsakymais.

1.2. Praktikos užduotis

- Vykdyti sistemos projektavimo veiklas.
- Vykdyti sistemos kūrimo veiklas.
- Vykdyti projekto apimties įvertinimą ir užduočių analizę.

1.3. Praktikos tikslas

Pritaikyti teorines ir praktines žinias apie programų sistemų kūrimą, įgytas Vilniaus universitete realiems projektams.

1.4. Spręsti praktikos uždaviniai

- Suprojektuoti duomenų bazės schemą.
- Išrinkti Java programavimo karkasą tinkamą debesijos kompiuterijai.
- Suprojektuoti duomenų sinchronizavimo architektūrą tarp atskirų duomenų bazių.
- Projektui parinkti kokybės užtikrinimo gerąsias praktikas ir jų vykdymo užtikrinimo būdus.

1.5. Praktinės veiklos planas

Praktinė veikla truko nuo 2021-09-01 iki 2021-11-30

1. Įvadas į projektą, naudojamus įrankius ir bendrą įmonės veiklą: 2021-09-01 iki 2021-09-10
2. Praktikos užduočių atlikimas: 2021-09-11 iki 2019-11-30

2. Įmonės apibūdinimas

2.1. Įmonės veiklos sritis

EPAM sistemos yra Baltarusijos ir Amerikos informacinių technologijų įmonės kuri specializuojasi užsakomaisiais projektais ir konsultavimu informacinių technologijų srityje.

2.2. Įmonės organizacinė struktūra

Įmonės padalinys Lietuvoje buvo įsteigtas prieš metus. Šiuo metu įmonėje dirba apie 300 darbuotojų ir šis skaičius vis auga. Didžioji dalis darbuotojų yra atvykusių iš Baltarusijos. Nors įmonė turi sąlyginai daug darbuotojų tačiau tik maža jų dalis būna ofise. Ofise dažniausiai būna apie 50 žmonių. Darbuotojai norėdami atvykti į ofisą turi užsirezervuoti darbo vietą vidinėje įmonės sistemoje, nes individualių darbo vietų nėra. Mano komandoje dirbo 6 žmonės.

2.3. Įmonės teikiamos paslaugos

1. Klienų programavimo projektų vykdymas - pagrindinė įmonės veiklos sritis yra klietų informacinių technologijų projektų vykdymas. Klientas ateina su prašymu atlikti projektą, EPAM įvertina kiek projektas kainuotų klientui ir pateikia pasiūlymą. Jeigu klientui tinka pasiūlymas ir nurodyti kaštai EPAM savo vidinėje sistemoje suranda reikiamos kompetencijos darbuotojus atlikti projektui. Jeigu reikia surasti darbuotojai praeiną darbo pokalbius pas klientą. Praėjus darbo pokalbius EPAM darbuotojas pradeda dirbti pas klientą. Pats EPAM darbuotojas gali nesutikti dirbti jam siūlomame projekte jeigu tas projektas jo nedomina.

2. Klientų konsultavimas informacinių technologijų klausimais - panašiom sąlygom kaip ir buvo minėta apie projektų vykgdymą, įmonė klientams teikia konsultavimo paslaugas, kuomet EPAM darbuotojas trumpą laiką atlieką kliento konsultavimą informacinių technologijų projektų klausimais. Konsultantas gali padėti įvertinti projekto apimtis, patarti kokių kompetencijų specialistų reikės, įvertinti reikiamą biudžetą ir panašius su projekto pradžia ar tolimesniu vystymu susijusiais klausimais.

2.4. Darbo sąlygos

Įmonė yra įsikūrusi Vilniuje Šeimyniškių g. 19-601. Pastatas yra patogioje miesto vietoje į kurią yra patogus susisiekimas viešuoju transpotu. Kompanija suteikia keletą nemokamo parkavimosi vietų, o kai jos pasibaigia netoliese yra nebrangios mokamos parkavimosi vietos. Ofise nėra asmeninių darbo vietų, todėl norint atvykti ir dirbti reikia užsirezervuoti darbo vietą vidinėje įmonės sistemoje. Įmonė leidžia dirbti iš namų, net ir šiuo metu kai karantino sąlygos leidžia dirbti iš ofiso. Per beveik metu kai dirbu šioje įmonėje ofise buvau tik keletą kartų ir jokių nusiskundimų dėl to neišgirdau. Darbuotojams dirbti iš namų yra suteikiama beveik visa reikalingą įranga(nešiojamas kompiuteris, monitorius, pelė ir kita). Pradėjęs dirbti įmonėje mėnesį praleidau vidiniame įmonės darbuotojų apmokymo procese, kuriame galėjau gintis savo programavimo žinias, bei buvau

apmokomas kaip sėkmingiau praeiti kliento darbo pokalbio procesą, kokių klausimų dažniausiai klausiama per darbo pokalbius ir kaip į juos atsakyti teisingai. Po mėnesio man buvo paskirtas projektas, kuriu darbo pokalbį sėkmingai praėjau ir pradėjau dirbti kliento projekte.

3. Praktikos veiklos aprašymas

3.1. Projektas

Aš dirbau prie Amerikos įmonės Vertex vienkartinio prisijungimo projekto. Įmonė Vertex specializuojasi mokesčių skaičiavimo programinės įrangos kūrime. Vertex parduodami produktai turi daug modulių į kuriuos galima prisijungti skirtingai būdais, su skirtingais prisijungimo duomenimis. Mūsų projekto tikslas buvo sukurti vienkartinio prisijungimo sistemą(angl. Single Sign-On) apjungiačia visus esamus prisijungimo būdus. Komandoje dirbo šeši žmonės: Viena projekto vadovė, keturi programuotojai ir vienas testuotojas.

3.2. Darbo procesas

Darbas šiame projekte vyko pagal Agile metodologiją. Kiento produkto savininkas(angl. product owner) arba sistemos architektas pateikdavo reikalavimus(angl. Epics) į užduočių valdymo platformą Atlasian. Mūsų komanda iš pateiktų reikalavimų sukurdavo istorijas(angl. Stories), kurias mes įvertindavome kiek laiko užtruktų atliktį kiekvieną istoriją. Kas dvi savaite mūsų komanda suplanuodavo sprintą(angl. Sprint) ir įvertindavo kiek užduočių galės atlikti per ateinančias dvi savaites. Po dviejų savaičių vienas komandos narys pristatydavo kokius darbus pavyko atlikti per dviejų savaičių sprintą ir pademonstruodavo progresą kliento projektų vadovei. Prieš kito sprinto planavimą komanda atlikdavo savo vidinę retrospektyvą ir padiskutuodavo kas buvo gerai ir ką reiktų tobulinti, kad ateities sprintai būtų produktyvesni. Sprinto metu komanda darbo trumpos penkiolikos minučių susitikimus su klientų papasakoti ką kiekvienas komandos narys radė praeitą dieną ir ką planuoją nueikti kitą dieną. Šio susitikimo metu taip pat išsprendžiamos problemos dėl dabartinio sprinto užduočių. Sprinto metu programuotojai atilikinėja suplanuotas užduotis ir atlikinėja kitų komandos narių kodo peržiūras. Testuotojas sprinto metu atlikinėja esamo funkcionalumo testavimą ir suradęs klaidą sukurią klaidos aprašą ir užduotį Atlasian sistemoje. Rastas klaida kode būna pataisoma esamo sprinto metu, įtraukiama į kito sprinto planavimą arba ignoruojama ir neatliekama. Kas atsitiks su rasta klaida nusprendžia pati komanda arba kliento architektas įvertinęs klaidos svarbumą ir kitas prioritetines užduotis.

3.3. Naudotos technologijos

Projekte buvo naudotas Micronaut 3 karkasas su Java 17 programavimo kalba. Duomenų sluoksniui kurti buvo pasirinkta reliacinė MySQL duomenų bazė ir objektų ryšių suvedimo biblioteka Hibernate. Programa buvo kuriama pagal REST API standartą. Aprašyti REST specifikacijai buvo naudojama OpenAPI specifikacijų kalba. Autorizacijai ir autentifikacijai mūsų komanda naudojo Auth0 platformą siekiant užtikrinti saugumą, palengvinti projekto įgyvendinimą ir sumažinti projekto biudžetą. Užtikrinti kodo kokybę mes naudojome įvarius statinio kodo analizės įrankius, tokius kaip: PMD, SpotBug, Checkstyle, Depedency Checker, Pom-lint. Visi šie statiniai kodo analizatoriai padėjo užtikrinti, kodo kokybę, tvarką ir saugumą.

3.4. Užduotis: Suprojektuoti duomenų bazės schemą.

3.5. Užduotis: Išrinkti Java programavimo karkasą tinkamą debesijos kompiuterijai.

3.6. Užduotis: Suprojektuoti duomenų sinchronizavimo architektūrą tarp atskirų duomenų bazių.

3.7. Užduotis: Projektui parinkti kokybės užtikrinimo gerąsias praktikas ir jų vykdymo užtikrinimo būdus.

Komanda nutarė, kad norint išlaikyti aukštą kodo kokybę ir sumažinti kodo klaidų ir saugumo spragų. Norint įgyventinti šia užduotį reikia surasti reikiamus įrankus padėsiančius užtikrinti užsibrėžtus tikslus bei integruoti juos į programos kūrimo procesą taip, kad būtų kuo sunkiau apeiti šiuos įrankius, nes gera programuotoju valia nevisuomet galima pasitikėti. Užduotį pradėjau vykdyti atlikdamas įvairių statinės analizės įrankių paiešką ir įvertinimą. Internete pavyko rasti nemažai mokamų ir nemokamų priemonių vykdančių kodo analizę.

3.7.1. JavaDocs validacija

Vienas iš būdų užtikrinti sklandų kodo palaikymą ateityje, kaip prie jo dirbs kiti programuotojai yra kodo dokumentacija. Dažnai projektuose nutinka taip, kad dokumentacija būna daroma atmestinai arba jos išvis nebūna ir kitai komandai perėmus projektą prireikia daug laiko ir pastangų toliau vystyti projektą. Dėl šių priežasčių komanda nutarė, kad dokumentacijos rašymas turi būti integruotas į programų kūrimo procesą. Vienas iš būdų Java kode rašyti dokumentacija yra naudoti Javadocs dokumentų generatorių kurio pagalba atitinkami Java kodo komentarai būtų sugeneruoti į dokumentaciją. Užtikrinti, kad JavaDocs komentarai būtų rašome kiekvienam viešam klasės metodui ir kontraktui naudosime maven-javadocs-plugin biblioteką, kuri užtikrina, kad visi komentarai būtų rašomi tinkamu formatu, visų reikiamų metodų ir kontraktų parametrai būtų nurodyti ir aprašyta ką daro kiekvienas iš metodų. Jeigu vienas iš nurodytų kiterijų nėra įgyventintas maven-javadocs-plugin parodo klaidą ir pasako ko trūksta kad dokumentacija būtų teisinga.

3.7.2. PMD

Gero kodo požymis yra kai jame laikomasi tvarkos: nėra nepanaudotų kintamųjų ar nenaudojamų bibliotekų, kitamųjų pavadinimai turi prasmę ir laikomasi jų pavadinimo standartų, kode nėra magiškų skaičių(angl. Magic number). Dažnai tokius dalykus galima pastebėti kodo peržiūros stadijoje, tačiau visi žmonės klysta ir kartais praleidžia šiuos dalykus pro pirštus, beto jeigu būtų įrankis galintis patikrinti šiuos dalykus kodo peržiūros užimtų mažiau laiko. Vienas iš atviro kodo įrankių plačiai naudojamos tokio tipo statiniai kodo analizei yra PMD. Jo pagalba galima patikrinti didelę aibę kodo stiliaus klaidų, gerųjų praktikų trūkumų bei daug kitų panašių dalykų. PMD turi savų trūkumų, kartais yra rodomos klaidos kurių negalima ištaisyti, pavyzdžiui rodomos klaidos

naudojant Lombok generuojamą kodą. Išspręsti tokius tūkumus padeda lanksti PMD konfigūracija, kurios pagalba galima išjungti norimas kodo analizės taisykles visam projektui arba tik vienam failui.

3.7.3. SpotBugs

SpotBugs yra statinės kodo analizės įrankis padedantis surasti kodo klaidas, tokias kaip neinicializuoti laukai, laukai turintys null reikšmę kai kode anotacijos null reikšmės neleidžia, sąlygos sakiniai visuomet gražinantys tą pačia reikšmę bei kitas klaidas kurios sintaksiškai yra teisingos, bet gali sukelti programos sutrikimus ateityje.

3.7.4. Kodo priklausomybių tikrinimas

Siekiant užtikrinti, kad projekto kodas būtų saugus svarbu užtikrinti, kad kode naudojamos priklausomybės būtų saugios. Šiam tikslui naudojome dependency-check-maven ir maven-enforcer-plugin bibleotekas, jos užtikrino, kad naudojamos kodo priklausomybių versijos neturi žinomų saugumo spragų ir priklausomybių versijos nesiduplikuoja arba nėra nurodytos dvi skirtingos versijos tai pačiai priklausomybei.

3.7.5. Checkmarx ir SonarQube

Kliento prašymu turėjome naudoti Checkmarx saugumo statinį analizatorių ir SonarQube statinių kodo analizatorių. Kaikurie dalykai analizuojami šių įrankių jau buvo daromi aukščiau minėtuose įrankiuose, bet buvo nutarta palikti naudojamus įrankius, nes jų vykdymo laikas yra trumpesnis ir perteklinė analizė nėra blogai.

3.7.6. Vieneto ir integraciniai testai

Dar vienas svarbus kodo kokybės aspektas yra testai. Nutarėme, kad projekto išeities kodas turi būti 80 procentų padengtas testais. Tą užtikrinti naudojome Jacoco biblioteką parodančia koks yra projekto kodo padengimo testais procentais.

3.7.7. Kodo formatavimas

Paskutinis aspektas kurį norėjome užtikrinti buvo kodo stiliaus pastovumas. Jeigu kodas visame projekte atrodo panašiai tampa lengiau skaityti ir suprasti kodą. Šiam tikslui pasirinkome plačiai naudojama Google Java style konvenciją, kurią tikrinom pagal Google suteiktą stiliaus tikrinimo įrankį checkStyle.

3.7.8. Integracija su Github Actions

Suprantome, kad reikia priversti programuotojus naudotis statinio kodo analizės įrankiais nes kitu atveju niekas jų nenaudos. Buvo nutarta šiuos įrankius sukonfigūruoti Github Action aplinkoje, kuri užtikrina, kad joks kodas negalėtų patekti į duomenų bazę prieš tai nepraėjęs visų minėtų

įrankių. Visi šie įrankiai kartu su Github Actions integracija yra toliau sėkmingai naudojami komandoje.

4. Rezultatai, išvados ir pasiūlymai

4.1. Rezultatai

- Suprojektuota ir įgyvendinta duomenų bazės schema.
- Įvertinti Java kalbos karkasai ir išrinktas karkasas leidžiantis greitesnį šaltą paleidimą(angl. Cold start) ir mažesnius failo dydžius.
- Sėkmingai suprojektuotas ir įgyvendintas duomenų sinchronizavimas tarp dviejų duomenų bazių.
- Išrinktos ir integruotos kodo analizės priemonės pagerinančios kodo kokybę ir projekto saugumą.
- Pagilinti darbo komandoje įgudžiai.
- Susipažinta su projektinių įmonių darbo praktika.

4.2. Išvados

Praktika įvyko sėkmingai. Jos metu išmokau naujų technologijų bei pagilinau žinias tose technologijose kurias jau mokėjau. Praktikos užduotys buvo įgyvendintos sėkmingai. Sėkmingai pritaikiau teorines ir praktinias žinias įgytas universitete. Darbe aprašytos užduotys buvo tik dalis per praktika nuveikto darbo ir žinių kurių pasisėmiau. Įgytas praktikos žinias ir toliau taikysiu savo darbe ir kituose ateities projektuose.

4.3. Privalumai ir trūkumai

Privalumai: praplėčiau savo žinias apie REST tipo projektus, modernius programavimo karkasus, statinius kodo analizatorius ir gerasias programavimo praktikas. Turėjau galimybę programuoti naujausia Java kalbos versija, kas yra retas malonumas.

Trūkumai: Projekto metu vykdavo labai daug susitikimų su klientu, kurie atitraukdavo dėmesį nuo darbo. Informacija iš visų šių susitikimų galėdavo būt perduodama išsiuntus elektroninį laišką vietoj pačio susitikimo.

4.4. Pasiūlymai

Siūlyčiau sumažinti susitikimų trukmę ir dažnumą. Būna tokių dienų kai visas dienos darbas būna dalyvauti susitikimuose, kurie net nėra susije su mūsų projekto darbu. Net jeigu susitikimai būna trumpi jie atitraukia dėmesį nuo užduočių darymo ir reikia laiko vėl susikaupti po susitikimo, kas mažina dienos produktyvumą.