PD.TVZ.HR
POLYTECHNIC & DESIGN Vol. 11, No. 1, 2023.
Mrežna inačica: ISSN 2459-6302
Tiskana inačica: ISSN 1849-1995
DOI: 10.19279/TVZ.PD.2023-11-1-04

KORISNIČKE PRIČE U RAZVOJU APLIKACIJE KROZ PRIMJENU U NASTAVI

USER STORIES IN APPLICATION DEVELOPMENT THROUGH TEACHING PROCESS

Vlatka Davidović, Ida Panev

Veleučilište u Rijeci, Vukovarska 58, 51000 Rijeka, Hrvatska

SAŽETAK

Korisničke priče se koriste u agilnom razvoju aplikacije kako bi opisale korisničke zahtjeve. Ukoliko su prevelike, teško ih je brzo implementirati. Stoga se one dijele u manje korisničke priče kojima se obuhvaća kompletna arhitektura sustava. Ovakva podjela aplikacijske arhitekture osigurava brz razvoj potpuno funkcionalnog dijela aplikacije koji odgovara zahtjevu korisnika i može se odmah isporučiti. U radu je opisan razvoj aplikacije na nastavi korištenjem tehnika raščlane korisničke priče te su predstavljene prednosti i nedostaci takvog pristupa.

Ključne riječi: *korisnička priča, agilni razvoj, web aplikacija, nastava*

ABSTRACT

User stories are used in agile application development to describe user requirements. If they are too large, it is difficult to implement them quickly. Therefore, they are divided into smaller user stories that encompass the complete system architecture. This division of the application architecture ensures the rapid development of a fully functional part of the application that meets the user's request and can be delivered immediately. This paper describes the development of the application in classes using the techniques of splitting of the user story. We present the advantages and disadvantages of such an approach.

Keywords: *user story, agile development, web application, teaching*

1. UVOD
1. INTRODUCTION

Brz razvoj i isporuka kvalitetnog softvera imperativ je s kojim se susreću timovi koji rade na njihovom razvoju. Prema istraživanjima, 70% organizacija koristi agilni pristup u razvoju softvera [1]. Agilni pristup karakteriziraju četiri osnovna principa: (1) pojedinci i njihovi međusobni odnosi su bitniji od procesa i alata, (2) softver koji radi je bitniji od opsežne dokumentacije, (3) suradnja s naručiteljem je bitnija od pregovaranja oko ugovora, (4) odgovor na promjene je bitniji od ustrajanja na planu [2]. Osnovni aspekti agilnog razvoja softvera su jednostavnost i brzina. Softver se razvija inkrementalno kroz niz kratkih iteracija i isporuka, pri čemu je bitna direktna i stalna komunikacija između korisnika i razvojnih inženjera te unutar razvojnog tima [3]. Ovaj pristup također stavlja u prvi plan naručitelja/korisnika i njegove zahtjeve. Čestim isporukama kvalitetnog softvera nastoji ga se držati zadovoljnim. Zahtjevi korisnika opisuju se korisničkim pričama.

Korisnička priča je kratak, jednostavan i neformalan opis jedne funkcionalnosti u aplikaciji koju treba napraviti i koja je vrijedna korisniku [4]. Za razliku od klasičnih scenarija u use-case prikazu, korisnička priča je manje formalna, općenitija, nije strogo određena i ostavlja prostor za daljnji dogovor s korisnikom oko detalja [5]. Korisničke priče ne bi smjele biti velike, inače se teško mogu obuhvatiti unutar kratkih razvojnih iteracija. Po potrebi se raščlanjuju na manje dijelove, kako bi se razbila kompleksnost zahtjeva te kako bi se razvoj softvera podijelio u manje funkcionalne cjeline. Takve cjeline se mogu lakše izgraditi.

Ovaj pristup korišten je na kolegiju Izgradnja objektno orijentiranih aplikacija na specijalističkom studiju Informatike Veleučilišta u Rijeci. Pred studente je postavljen zadatak da izgrade web aplikaciju koja bi se koristila u nastavi na odjelu Održivog agroturizma Veleučilišta u Rijeci. Kroz nastavu su isprobani neki elementi agilnog pristupa u razvoju softvera poput raščlanjivanja korisničkih priča, timskog rada, transparentnosti i vođenih sastanaka, kako bi se brzo i u manjim iteracijama isporučila funkcionalna aplikacija. Svrha ovakvog načina rada bilo je sprječavanje situacije u kojoj studenti ne bi uspjeli dovršiti aplikaciju zbog nedostatka vremena.

U ovom radu je stavljen naglasak na korisničke priče preko kojih su napravljeni manji zadaci koji nisu bili teški za implementaciju, a studenti su ih mogli razvijati paralelno svaki svoj te ih kasnije spojiti i testirati. Kroz praktičan primjer predstavljen je jedan način podjele korisničke priče koji se može primijeniti u nastavi tako da se tijekom jednog semestra može razviti i isporučiti korisna aplikacija.

2. KORISNIČKA PRIČA U AGILNOM RAZVOJU APLIKACIJA

2. USER STORY IN AGILE SOFTWARE DEVELOPMENT

Gledano sa strane krajnjeg korisnika, korisnička priča daje odgovor na pitanje zašto se neki dio aplikacije ili funkcionalnosti mora izraditi, izmijeniti ili popraviti. Sastoji se od tri glavna elementa: tko - označava aktera ili osobu koja zahtijeva neki krajnji cilj, što - označava ono što akter želi postići kao krajnji cilj na kraju radnje, zašto – opisuje razlog zbog kojeg postoji korisnička priča.

Kao (tko?) želim (što?) kako bih (zašto?)

Za svaku korisničku priču postoji kriterij prihvatljivosti na temelju kojeg se izrađuju podržani modeli. Kriteriji prihvatljivosti se odnose na skup unaprijed definiranih zahtjeva koji trebaju biti zadovoljeni kako bi korisnička priča bila označena kao završena.

Kvaliteta korisničke priče može se mjeriti kroz INVEST (*Independent, Negotiable, Valuable, Estimable, Small, Testable*) kriterije. Prema njima korisničke priče bi trebale biti međusobno neovisne, tako da se mogu implementirati po bilo kojem redoslijedu. Ne bi trebale biti vezane uz neku specifičnu značajku ili detalje te su otvorene za dogovor. Trebale bi nositi vidljivu vrijednost krajnjem korisniku. Može se procijeniti trajanje implementacije, pri čemu se veće i općenitije priče teže mogu procijeniti. Također, jedna priča bi trebala biti dovoljno malena kako bi se uklopila u iteraciju. I konačno, može se testirati. Time je implementacija korisničke priče završena [6].

Kriterije mogu definirati razvojni okviri poput *Agile Requirements Verification Framework* koji promatra kompletnost, jednoobraznost te konzistentnost i korektnost korisničke priče [7], QUS (*Quality User Story*) gdje je fokus na dobro formiranoj, nedjeljivoj, minimalnoj, konceptualnoj, problemski orijentiranoj, jednoznačnoj, potpuno objašnjenoj i procjenjivoj korisničkoj priči [8] te USQM (*User Story Quality Measurement*) koji ima tri nivoa provjere kvalitete korisničke priče: osnovnu, upravljačku i provjeru prihvatljivosti [9].

Ovi kriteriji posredno utječu na to na koji način, do koje mjere i veličine raščlanjivati korisničku priču. Veličina korisničke priče nije strogo specificirana, ali se kao jednostavna mjera može uzeti razdioba korisničke priče na dijelove koje je moguće u nekoliko dana izgraditi i testirati [10], pa se tom mjerom vodilo kod razdiobe korisničkih priča na nastavi.

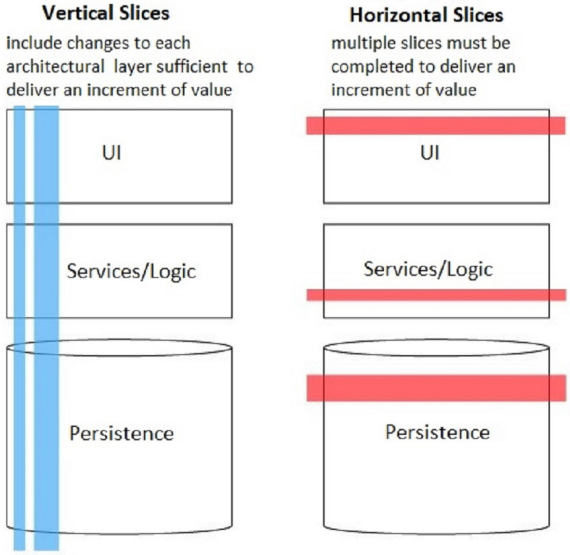
3. VERTIKALNA APLIKACIJSKA ARHITEKTURA

3. VERTICAL APPLICATION ARCHITECTURE

Arhitektura neke aplikacije vezana je uz organizaciju komponenti u aplikaciji kao i način komuniciranja tih komponenti. Tri su glavna aplikacijska sloja unutar slojevitog arhitekturnog stila: prezentacijski sloj, poslovni sloj i sloj pristupa podacima. Odnosno, korisničko sučelje, logika i pohrana podataka. Ovi slojevi su horizontalni i svaki dio aplikacijskog sloja ima svoje specifične odgovornosti [11]. Razvoj

aplikacije može ići prema arhitekturnim slojevima odnosno horizontalno. To znači da se zasebno razvija korisničko sučelje, zasebno sloj pristupa podacima i zasebno poslovna logika. U razvoju aplikacije to znači da će se svaki sloj razvijati kompletno, a tek onda isporučiti. Kod agilnog pristupa aplikacija se dijeli u vertikalne segmente i tako obuhvaća sve slojeve arhitekture, ali u malim funkcionalnim povezanim dijelovima (slika 1). To ujedno znači da će se razvijati jedan funkcionalni segment aplikacije i da će morati obuhvatiti sve slojeve.

Vertikalna segmentacija se radi tako da odgovara jednoj korisničkoj priči, a ukoliko je neka priča općenita i prevelika, ona se dijeli na manje [12].



Slika 1 Horizontalno i vertikalno dijeljenje slojeva arhitekture aplikacije [13]

***Figure 1** Horizontal and vertical slicing through architectural application layers [13]*

Razbijanje korisničke priče se može napraviti na više načina [10][14], do razine zadovoljavajuće kvalitete.

Jednostavan način za podjelu korisničkih priča u manje je korištenjem SPIDR (*Spike, Paths, Interface, Data, Rules*) tehnika. Gotovo svaka korisnička priča može se pojednostaviti korištenjem jedne od ovih 5 tehnika:

- *Spike* – kreiraju se mali prototipovi jednog dijela funkcionalnosti koji se obično koriste za provjeru izvedivosti kod nove tehnologije.
- *Paths* – kreiranje puteva u korisničkim pričama. Tamo gdje postoji račvanje u pričama, mogu se definirati odvojene korisničke priče.
- *Interface* – kreiranje korisničke priče za određeni uređaj (web, mobilna, desktop aplikacija). Kod kompleksnog razvoja, odvaja se razvoj za najčešće korišteni tip uređaja, a ostali idu u kasniji razvoj.
- *Data* – priča temeljena na najčešće korištenom skupu podataka te ne uključuje odmah sve opcije koje postoje.
- *Rules* – priča slijedi poslovno pravilo ili tehnološki standard [15].

4. KORISNIČKE PRIČE U WEB-APLIKACIJI „KVIZ O BILJNIM VRSTAMA“

4. USER STORIES FOR WEB APPLICATION „PLANTS SPECIES QUIZ“

Zadatak koji su dobili studenti je izrada web-aplikacije „Kviz o biljnim vrstama“ koja bi se koristila za učenje i provjeru znanja o biljnim vrstama. Korisnicima te aplikacije bi se nasumično generirala pitanja. Za svako pitanje korisnik može odabrati jedan ili više mogućih odgovora te mu aplikacija odmah daje povratnu informaciju o točnim/netočnim odgovorima. Ukoliko je odgovor netočan, korisniku se prikaže i točan odgovor. Završavanjem kviza prikazuju se i ukupni rezultati. Rezultati se ne trebaju spremati, nego su informativnog karaktera.

Uz korisnika aplikacije vezana je sljedeća terminologija, koju on treba razumjeti:

- Glavni pojam na kojeg se vežu svi podaci je biljna vrsta.
- Biljna vrsta - organizam koji uz čovjeka i životinje tvori živi svijet.
- Rod – ime roda predstavlja skupinu biljnih vrsta.
- Botanička porodica – kategorija koja obuhvaća jedan ili više srodnih biljnih rodova.

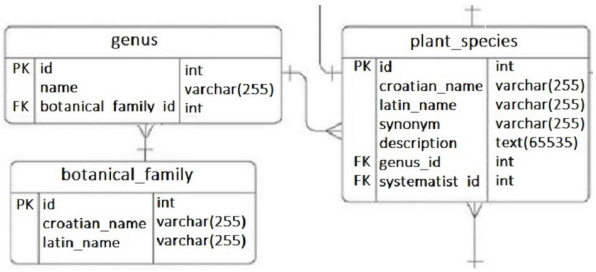
Zahtjevi su dani u sljedećim kratkim specifikacijama:

- Treba osigurati da se korisnicima aplikacije nasumično prikazuju 3-4 tipa pitanja. Ovisno o tipu pitanja može se ponuditi više nasumično odabranih odgovora od kojih je jedan ili više odgovora točno.
- Nakon odabira odgovora, osigurati povratnu informaciju te prikaz ispravnih odgovora.
- Aplikacija na kraju daje zbirni prikaz točnih i netočnih odgovora.

Tehnički zahtjevi:

- Aplikacija treba koristiti podatke koji se već nalaze u postojećoj bazi podataka.

Na slici 2 je prikazan dio relacijskog dijagrama baze podataka iz koje treba dohvatiti potrebne podatke.



Slika 2 *Dio relacijskog dijagrama aplikacije Kviz o biljnim vrstama*

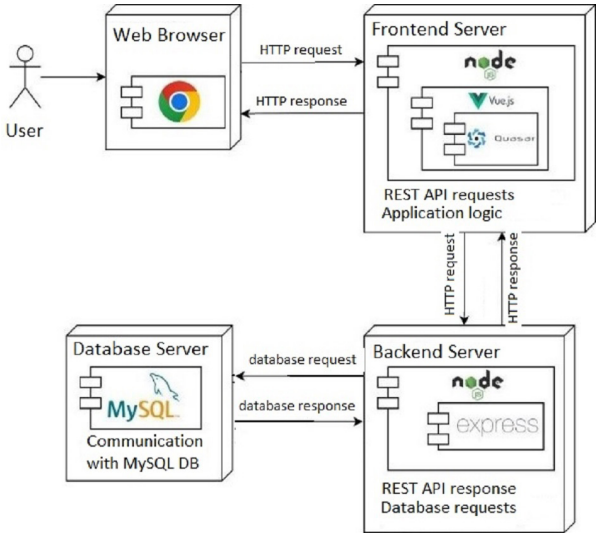
Figure 2 *Part of relational diagram for application Plants Species Quiz*

5. ARHITEKTURA WEB-APLIKACIJE „KVIZ O BILJNIM VRSTAMA“

5. ARCHITECTURE OF WEB APPLICATION „PLANTS SPECIES QUIZ“

Arhitektura web-aplikacije „Kviz o biljnim vrstama“ podijeljena je na tri nezavisne cjeline: klijentska strana (engl. *frontend*), poslužiteljska strana (engl. *backend*) i baza podataka (engl. *database*).

- S klijentske strane potrebno je kreirati prozore za prikaz pitanja i interakciju s korisnikom te formirati upite prema serverskoj strani.
- Sa serverske strane potrebno je kreirati API koji će na temelju zahtjeva od strane klijenta slati pitanja i odgovore.



Slika 3 *Arhitektura sustava Kviz o biljnim vrstama*

Figure 3 *System architecture for Plant Speies Quiz*

Za razvoj klijentskog dijela aplikacije odabran je Quasar.js razvojni okvir. Quasar.js je napravljen na razvojnom okviru Vue.js. Datoteke se sastoje od tri dijela: template, script i style. U <template> dijelu .vue datoteke postavlja se struktura grafičkog sučelja koja će se prikazati unutar web preglednika. Quasar.js ima vlastite grafičke komponente koje se koriste za interakciju s korisnikom: labele, liste, gumbe, polja za unos teksta, slike, tablice i slično. U <script> dijelu datoteke se nalazi JavaScript programski kod, u kojem se može smjestiti aplikacijska logika. Klijentski dio aplikacije isporučuje web-stranicu na zahtjev web-preglednika (slika 3).

Poslužiteljski dio aplikacije napravljen je u Node.js i Express.js razvojnom okviru. Node.js je razvojno okruženje za samostalno izvršavanje JavaScript aplikacija, između ostalih i web poslužitelja ukoliko se pokreće na poslužiteljskoj strani (engl. *server-side*). Express.js je poslužiteljski razvojni okvir za izgradnju REST API-ja. Sadrži upravljanje putanjama (engl. *routing*) za pristup resursima, srednji sloj aplikacije (engl. *middleware*) za kontrolu i upravljanje poslovnom logikom te poglede (engl. *view*). Uloga poslužiteljskog dijela aplikacije je omogućiti komunikaciju između baze podataka i klijentskog dijela aplikacije putem REST arhitekture. Prije implementacije krajnjeg rješenja potrebno je definirati priključne točke koje će klijentska strana koristiti za dohvat podataka. Priključna točka sadrži URI (engl. *Uniform*

Resource Identifier) resurs i jednu od HTTP metoda: GET, POST, PUT, PATCH, DELETE.

6. PODJELA KORISNIČKIH PRIČA NA ZADATKE

6. DIVIDING USER STORIES INTO TASKS

Korisnička priča je podijeljena na manje dijelove korištenjem SPIDR tehnike: kreiran je prototip za jedan tip pitanja, koji je trebao poslužiti kao primjer prema kojem se mogu napraviti ostala pitanja.

Glavna korisnička priča:

Kao student želim provjeru znanja o biljnim vrstama kako bih se spremio za ispit.

Izdvojen je jedan primjer za izradu prototipa:

Kao student želim provjeru znanja o raspoznavanju *botaničke porodice* ukoliko je dana *biljna vrsta*, kako bih provjerio znam li *biljnu vrstu* svrstati u *botaničku porodicu*.

Primjer je raščlanjen na manje ovisne dijelove, te je za svaki dan kriterij prihvatljivosti:

1. Kao student želim da mi se nasumično generira pitanje o raspoznavanju *botaničke porodice* ukoliko je dana *biljna vrsta*, kako bih provjerio znam li *biljnu vrstu* svrstati u *botaničku porodicu*.

Kriterij prihvatljivosti: Prikazano je pitanje: Kojoj *botaničkoj porodici pripada biljna vrsta (naziv)*?

2. Kao student želim da mi se prikaže više mogućih nasumično odabranih *botaničkih porodica* te jedna točna, kako bih mogao prepoznati i odabrati točan odgovor.

Kriterij prihvatljivosti: Ponuđeno je više odgovora od kojih je jedan točan.

3. Kao student želim da su ponuđene *botaničke porodice* nasumično izmiješane kako bih izbjegao da zapamtim redoslijed odgovora.

Kriterij prihvatljivosti: Odgovori su svaki put nasumično izmiješani.

4. Kao student želim da mi se na temelju mog odgovora ponudi povratna informacija točno/netočno kako bih znao jesam li dobro odabrao *botaničku porodicu*.

Kriteriji prihvatljivosti: Odabrani odgovor se provjerava.

Osigurana je povratna informacija: TOČAN/NETOČAN odgovor.

5. Kao student želim da mi se povratno da točan odgovor ukoliko sam odgovorio/la krivo, kako bih ponovio/la odnos između dane *biljne vrste* i *botaničke porodice*.

Kriterij prihvatljivosti: Ispisuje se točan odgovor.

Na temelju korisničke priče napravljeni su zadaci koji imaju SMART (*Specific, Measurable, Achievable, Relevant, Time-boxed*) karakteristike. Moraju biti specifični, mjerljivi, dostignuti, relevantni i vremenski ograničeni [16].

Za 1. dio razrađeni su sljedeći zadaci:

Na klijentskom dijelu:

- Predložiti grafičko rješenje za postavljanje pitanja i davanje odgovora te davanje povratne informacije studentu.
- Na temelju grafičkog prijedloga naći i predložiti grafičke komponente iz Quasar.js razvojnog okvira .

Na poslužiteljskom dijelu:

- Izraditi web servis (API) za dohvat svih biljnih vrsta iz baze.

Aplikacijska logika:

- Poziv API-ja za dohvat svih biljnih vrsta.
- Nasumično odabrati jednu biljnu vrstu iz popisa svih biljnih vrsta.

7. IMPLEMENTACIJA ZADATAKA IZ KORISNIČKE PRIČE

7. IMPLEMENTATION OF THE TASKS FROM THE USER STORY

Na klijentskom dijelu predložena su grafička rješenja od kojih je jedno odabrano i implementirano u Quasar.js razvojnom okviru.

Da bi se pitanje moglo prikazati, treba ga dohvatiti iz baze. Na poslužiteljskom dijelu, korištenjem Node.js i Express razvojnog okvira razvijen je API za dohvat svih biljnih vrsta iz baze (slika 4). API ima priključnu točku preko koje se može pozvati i dostaviti podatke. Poziva se s

klijentskog dijela aplikacije HTTP GET metodom i prosljeđuje joj sve biljne vrste u JSON zapisu.

```
// Retrieve all plant_species
app.get('/plant_species', function (request, response) {
    dbConn.query('SELECT * FROM plant_species',
                 function (error, results, fields) {
        if (error) throw error;
        return response.send({
            error: false, data: results,
            message: 'plant_species_list.'
        });
        });

    module.exports = app;
```

Slika 4 *API za dohvat biljnih vrsta*

Figure 4 *API for retrieving plant species*

Logika aplikacije nalazi se unutar Quasar.js razvojnog okvira. Potrebno je napraviti poziv API-ja za dohvat svih biljnih vrsta, zatim nasumično odabrati jednu (slika 5) i predstaviti je u pitanju.

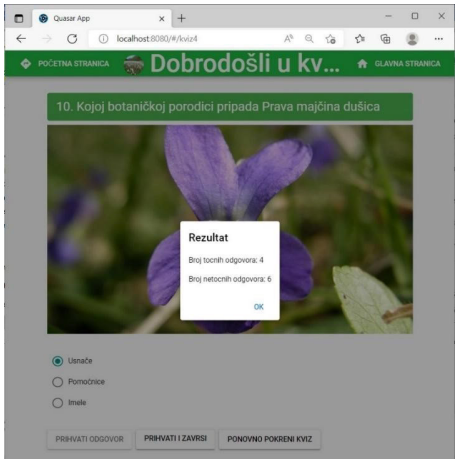
```
// funkcija koja dohvaca random plant species i postavlja
// vrijednost u state.plant
async function randomPlant() {
  const jsonObject = await axios.get(
    `http://localhost:3000/plant_species/`
  );
  let randomPlant =
    jsonObject.data.data[
      Math.floor(Math.random() * jsonObject.data.data.length)
    ];

  // plant se sprema u state.plant
  state.plant = randomPlant;
```

Slika 5 *JavaScript funkcija za nasumični odabir biljne vrste*

Figure 5 *JavaScript function for random plant selection*

Na sličan način su rješavani svi ostali zadaci iz korisničkih priča. Aplikacija je napravljena do verzije koja se može koristiti (slika 6).



Slika 6 *Web aplikacija Kviz o biljnim vrstama*

Figure 6 *Web application Plant Species Quiz*

8. RAZVOJ APLIKACIJE TIJEKOM NASTAVNOG PROCESA

8. APPLICATION DEVELOPMENT DURING COURSE

Na razvoju aplikacije radio je tim od 10 studenata. Na početku semestra upoznati su s osnovama agilnog pristupa, zadatkom i tijekom rada na projektu. Dane su upute za timski rad na projektu. Kroz raščlanjivanje korisničkih priča dobiveni su zadaci koji su podijeljeni studentima. Nastava je korištena za rješavanje grešaka i zastoja, davanje smjernica, timski rad i zajedničko rješavanje otvorenih pitanja. Na nastavi su studenti dobivali zadatke za daljnje rješavanje. Studenti su poticani da rade u dijeljenom okruženju. Tijekom rada studenti su se prvi put sreli s Quasar.js, Node.js i Express razvojnim okvirima. Najviše vremena je utrošeno na kreiranje aplikacije prema prvoj korisničkoj priči (5.-9. tjedan), u što spada učenje razvojnih okvira i prilagođavanje postavljenoj kulturi rada u timu (tablica 1). U nadogradnji aplikacije napravljene su još dvije iteracije isporuke.

Tablica 1 *Vrijeme utrošeno na pojedine dijelove razvoja*

Table 1 *Time spent on individual parts of development*

Vremensko trajanje	**Rad na pojedinim dijelovima projekta**
1. - 4. tjedan	Upoznavanje s agilnim pristupom, zadatkom, davanje smjernica za korištenje sustava za upravljanje izvornim kodom i priprema za timski rad
5. - 9. tjedan	Podjela korisničkih priča, rješavanje zadataka prema korisničkim pričama, testiranje, isporuka
10. - 13. tjedan	Nadogradnja aplikacije, testiranje, isporuka

9. PREDNOSTI I OGRANIČENJA
9. ADVANTAGES AND LIMITATIONS

Kroz ovaj pristup studenti su relativno brzo su vidjeli rezultate čime su bili dodatno motivirani za rad. Također je podjela zadataka napravljena tako da svaki student praktično upozna cjelokupnu arhitekturu sustava i implementira jedan dio. Studenti se fokusiraju na izradu manje cjeline i mogu usput učiti tehnologiju. Obzirom da im nije trebalo dugo za završiti neki zadatak, mogli su ostati fokusirani na njega i ne kasniti s njegovim izvršavanjem. Manje zadatke su mogli riješiti u kraćem vremenskom periodu i s manje truda nego da su dobili kompleksniji problem te su se na taj način iteracije u razvoju mogle brže odraditi.

S druge strane, studenti nisu sudjelovali u formiranju i raščlanjivanju korisničkih priča nego im je tehnika opisana na početku semestra. Sudjelovali su samo na raščlani priča u manje zadatke kako bi bolje razumjeli zadatak i tehnologiju. U ovom slučaju studenti su ostali zakinuti za rješavanje kompleksnijih problema i učenje ove tehnike raščlane korisničkih priča koja bi im mogla pomoći kod izrade budućih projekata. Ovaj dio ostaje za buduću prilagodbu u nastavi.

10. ZAKLJUČAK
10. CONCLUSION

Korisničke priče su samo dio agilnog pristupa u razvoju aplikacija, no izuzetno su bitne. S korisničkim zahtjevima počinje razvoj aplikacije te se cijeli proces razvoja i konačne isporuke naslanja na njih. Vertikalnim raščlanjivanjem korisničkih priča na manje cjeline omogućen je brz razvoj aplikacije. Dosadašnje iskustvo u radu na ovakvim projektima u nastavi pokazalo je da se događaju kašnjenja u razvoju radi kompleksnijih dijelova koje pojedinci nisu mogli ili stizali na vrijeme razviti, čime bi cijeli projekt kasnio. Međutim, ukoliko se stavi fokus na razvoj pojedinih manjih dijelova, moguće je razviti i isporučiti funkcionalnu aplikaciju tijekom semestra. Ovaj pristup se može iskoristiti na tipu grupnog projekta u nastavi, ako je cilj razvoj i isporuka aplikacije, no mogućnosti su i šire. U tom dijelu postoji potencijal za daljnje istraživanje.

11. REFERENCE
11. REFERENCES

[1.] Gartner; Results Summary: Agile in the Enterprise; 2019. https://circle.gartner.com/Portals/2/Resources/pdf/Agile%20in%20the%20Enterprise%202019%20-%20Results%20Summary%20(updated).pdf [Pristupljeno: 11.2022.]

[2.] Williams, L.; Cockburn, A.; 2003. Agile software development: It's about feedback and change. Computer, 36(6), pp.39-43; Print ISSN: 0018-9162 , Electronic ISSN: 1558-0814

[3.] Baham, C.; Hirschheim, R.; 2022. Issues, challenges, and a proposed theoretical core of agile software development research. Inf Syst J.; 32: 103– 129. doi:10.1111/isj.12336

[4.] Cohn, M.; 2004. User Stories Applied: For Agile Software Development; Addison-Wesley Professional; ISBN 0-321-20568-5

[5.] Spurrier, G.;Topi, H.; 2022. Synergistically Employing User Stories and Use Cases in the Practice and Teaching of Systems Analysis and Design. Communications of the AIS, p.561, ISSN: 1529-3181

[6.] Buglione, L.; Abran, A.; 2013. October. Improving the user story agile technique using the invest criteria. In 2013 Joint Conference of the 23rd International Workshop on Software Measurement and the 8th International Conference on Software Process and Product Measurement (pp. 49-53). IEEE; doi: 10.1109/IWSM-Mensura.2013.18

[7.] Heck, P. and Zaidman, A., 2014. A Quality Framework for Agile Requirements: A Practitioner's Perspective. Technical Report Series TUD-SERG-2014-006. ISSN: 1872-5392

[8.] Lucassen, G.; Dalpiaz, F.; van der Werf, J.M.E.M. et al.; 2016. Improving agile requirements: the Quality User Story framework and tool; Requirements Eng 21, pp.383–403. doi: 10.1007/s00766-016-0250-x

[9.] Lai, S.T.; 2017. A user story quality measurement model for reducing agile software development risk; International Journal of Software Engineering &

Applications, 8(2), pp.75-86; doi: 10.5121/ijsea.2017.8205

[10.] Patton, J.; 2014. User Story Mapping, O'Reilly Media, Inc.; ISBN: 978-1-491-90490-9

[11.] Richards, M.; Ford, N.; 2020. Fundamentals of Software Architecture; O'Reilly Media, Inc; ISBN: 978-1-492-04345-4

[12.] Ratner, I. M.; Harvey, J.; Vertical Slicing: Smaller is Better; 2011. AGILE Conference; doi:10.1109/agile.2011.46

[13.] Richard Lawrence; Peter Green; The Humanizing Work Guide to Splitting User Stories; https://www.humanizingwork.com/the-humanizing-work-guide-to-splitting-user-stories/#why-story-splitting-matters; [Pristupljeno: 11.2022.]

[14.] Adzic, G.; Evans, D., Roden, T., 2014. Fifty Quick Ideas to Improve Your User Stories; Neuri Consulting LLP, London, UK; ISBN: 978-0-9930881-0-0

[15.] Mike Cohn; Split Stories Using SPIDR; https://www.mountaingoatsoftware.com/exclusive/spidr-poster-download; [Pristupljeno: 11.2022.]

[16.] Pugh, K.; 2010. Lean-agile acceptance test-driven development: better software through collaboration; Pearson Education; ISBN: 978-0-321-71408-4

AUTORI · *AUTHORS*

• Vlatka Davidović - viši predavač na Veleučilištu u Rijeci. Završila je studij matematike i informatike u Rijeci. Područje interesa joj je programsko inženjerstvo s naglaskom na razvoj aplikacija. Predaje na kolegijima objektno orijentirane tehnologije, izgradnja objektno orijentiranih aplikacija, razvoj web aplikacija. Doktorandica je na Fakultetu informatike i digitalnih tehnologija gdje se bavi područjem umjetne inteligencije.

Korespondencija · *Correspondence*
vlatka.davidovic@veleri.hr

• Ida Panev - viši predavač na Veleučilištu u Rijeci. Doktorirala je na Filozofskom fakultetu u Zagrebu na području informacijsko komunikacijskih tehnologija. Područje interesa su joj razvoj informacijskih sustava i utjecaj tehnologije u informatičkom obrazovanju. Predaje na kolegijima razvoj informacijskih sustava, modeliranje podataka i procesa, baze podataka, multimedijski sustavi, uvod u informacijsko komunikacijske tehnologije.

Korespondencija · *Correspondence*
ida.panev@veleri.hr