Web aplikacija za vrednovanje digitalnih obrazovnih igara

Petrović, Ana

Fakultet informatike i digitalnih tehnologija,

Sveučilište u Rijeci

Web aplikacija za vrednovanje digitalnih obrazovnih igara

Diplomski rad

Mentorica: prof. dr. sc. Nataša Hoić-Božić

Izradila: Ana Petrović

Smjer: Informacijsko komunikacijski sustavi

Rijeka, rujan 2022.

FAKULTET INFORMATIKE I DIGITALNIH TEHNOLOGIJA
SVEUČILIŠTE U RIJECI

UNIRI

Rijeka, 1.7.2022.

Zadatak za diplomski rad

Pristupnik: Ana Petrović

Naziv diplomskog rada: Web aplikacija za vrednovanje digitalnih obrazovnih igara

Naziv diplomskog rada na eng. jeziku: Web application for evaluating digital educational games

Sadržaj zadatka:

Konceptualni okvir za promišljanje o dizajnu korisničkog sučelja (UI) i dizajnu korisničkog iskustva (UX) ili model elemenata UI/UX može se predstaviti kroz pet ravnina ili faza: površina (*Surface*), kostur (*Skeleton*), struktura (*Structure*), opseg (*Scope*) i strategija (*Strategy*). Proces dizajniranja web aplikacije uključuje promišljanje o aspektima svih ovih faza, od strategije do vizualnog predstavljanja aplikacije u fazi površine.

Zadatak završnog rada je opisati proces dizajniranja web aplikacije prema modelu elemenata UX/UI na primjeru vlastite web aplikaciju za vrednovanje digitalnih obrazovnih igara. Opisati će se čitavi postupak dizajniranja i izrade te kao praktični dio rada razviti web aplikacija prema pripremljenom UX/UI dizajnu. *Wireframe* i *mockup* koji prikazuju vizualni dizajn web aplikacije te prototip bit će izrađeni u alatu Figma, a web aplikacija će biti programirana korištenjem Microsoft ASP.NET Core i React JS razvojnih okvira.

Mentor:
Prof. dr. sc. Nataša Hoić-Božić

Voditeljica za diplomske radove:
Prof. dr. sc. Ana Meštrović

Zadatak preuzet: 1.7.2022.

(potpis pristupnika)

Fakultet informatike i digitalnih tehnologija
Radmile Matejčić 2, 51000 Rijeka, Hrvatska
http://inf.uniri.hr ured@inf.uniri.hr
Tel: +385 (0)51584700 | Fax: +385 (0)51 584 749

UNIRI | YUFE

Zahvala

Zahvaljujem mentorici prof. dr. sc. Nataši Hoić-Božić na podršci i korisnim savjetima tijekom pisanja i izrade ovog rada. Posebnu zahvalu dugujem svojoj obitelji na strpljenju i velikoj podršci tijekom cijelog studiranja.

Sadržaj

Sažetak

Cilj ovog diplomskog rada je dizajnirati i razviti aplikaciju za vrednovanje digitalnih obrazovnih igara. Zadatak rada je opisati proces dizajniranja vlastite web aplikacije prateći korake modela elemenata UX/UI dizajna. *Wireframe* i *mockup* vizualni prikazi aplikacije te prototip izradit će se u alatu Figma. Web aplikacija bit će programirana korištenjem Microsoft ASP .NET Core i ReactJS razvojnih okvira. U prvoj cjelini biti će objašnjeni svi koraci dizajniranja aplikacije za vrednovanje digitalnih igara, a u drugoj najvažniji dijelovi koda koji su bili potrebni za njen razvoj.

Ključne riječi:

UI/UX dizajn, *wireframe*, *mockup*, Figma, ASP .NET Core, ReactJS, web aplikacija, digitalne obrazovne igre

1. Uvod

Dizajn korisničkog sučelja (UI) odnosi se na cjelokupnu interakciju koju korisnik ima sa sučeljem, dok dizajn korisničkog iskustva (UX) uključuje emocije, stavove i ponašanja koja korisnik doživljava tijekom uporabe proizvoda. Model elemenata UI/UX može se predstaviti kroz pet ravnina, a to su: površina (*Surface*), kostur (*Skeleton*), struktura (*Structure*), opseg (*Scope*) i strategija (*Strategy*). U diplomskom radu je model elemenata UI/UX pojašnjen dizajniranjem i razvojem vlastite web aplikacije za vrednovanje digitalnih obrazovnih igara. Tema aplikacije je iz područja učenja temeljenog na igri (eng. *Game-Based Learning* – GBL), a odabrana je jer se digitalne igre danas sve više koriste za učenje i poučavanje te ova aplikacija ima mogućnost daljnjeg proširivanja i korištenja od strane nastavnika.

GBL je pristup koji uključuje korištenje digitalnih obrazovnih igara čija je svrha ostvarivanje određenih ishoda učenja. Danas se one sve više koriste za motiviranje učenika, povećanje njihova angažmana te poboljšanja samih rezultata učenja. Korištenjem digitalnih obrazovnih igara kod učenika je moguće razvijati vještine komunikacije i suradnje te informacijsku i digitalnu pismenost već od nižih razreda osnovne škole. Kako bi se potaknulo što češće korištenje ovakvog načina učenja, aplikacija za vrednovanje digitalnih obrazovnih igara pomoći će nastavnicima u pronalaženju odgovarajućih igara za njihove učenike.

Backend aplikacije za vrednovanje digitalnih obrazovnih igara napravit će se korištenjem ASP.NET Core besplatnog web okvira otvorenog koda, kojeg je razvila tvrtka Microsoft. *Frontend* aplikacije kodirat će se u programskom jeziku Javascript korištenjem ReactJS razvojnog okvira. Dizajn je potrebno izraditi slijedeći korake UX i UI dizajna. Prije početka kodiranja potrebno je osmisliti strategiju za korištenje aplikacije, napisati funkcionalne zahtjeve i zahtjeve sadržaja te osmisliti dijagram arhitekture koji prikazuje strukturu aplikacije za vrednovanje digitalnih igara. U sljedećoj fazi potrebno je prikazati *wireframe*-ove, *mockup* prikaze, *moodboard* te stilski vodič za aplikaciju. Krajnji korak prije početka kodiranja je izrada prototipa u Figmi koji prikazuje kako će aplikacija funkcionirati.

Aplikacija za vrednovanje digitalnih obrazovnih igara omogućit će korisnicima, najčešće nastavnicima, korištenje nekoliko funkcionalnosti. Registriranje u aplikaciju bit će besplatno te će se nastavnik moći prijavljivati kako bi mogao pregledavati igre. Igre će biti prikazane u obliku kartica podijeljene na nekoliko stranica te će postojati mogućnost filtriranja radi lakšeg pretraživanja željenog područja i sl. Za odabranu igru korisnici će moći dati komentar i ocjenu kao i pročitati mišljenja nastavnika koji su ranije pregledavali i odigrali igru. Za svaku igru navest će se i dodatne informacije te, ako su dostupne, i video upute.

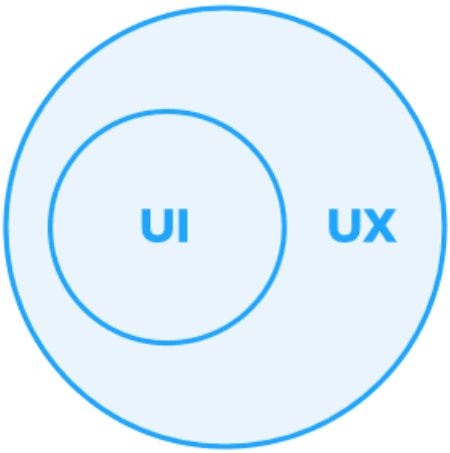
U radu će se opisati dizajn i razvoj aplikacije za vrednovanje digitalnih obrazovnih igara. U prvom dijelu će biti objašnjeni koraci UX i UI dizajna, a zatim sama izrada aplikacije u odabranim tehnologijama. U zaključku će biti rezimirane sve važnije spoznaje o obrađenoj temi do kojih se došlo prilikom izrade ovog rada.

2. Dizajniranje aplikacije

Prije početka izrade (kodiranja) aplikacije važno je dobro isplanirati dizajn vodeći računa o korisničkom sučelju, ali i o korisničkom iskustvu. Riječ je o različitim aspektima procesa dizajniranja te je važno objasniti razliku između ovih dvaju područja.

Dizajn korisničkog sučelja (eng. *user interface - UI)*) odnosi se na vizualne elemente digitalnog proizvoda, fokusira se na površinu, kako produkt izgleda i funkcionira, dok se dizajn korisničkog iskustva (eng. *user experience - UX)* odnosi na cjelokupno iskustvo koje korisnik ima tijekom interakcije s proizvodom, npr. na njegove emocije odnosno osjećaj zadovoljstva ili lošeg iskustva [2]. UI dizajn je stvarni dizajn sučelja nekog proizvoda (za web sjedište fokusiran na funkcionalnu organizaciju stranice i specifične elemente (gumbi, linkovi, izbornici,…) koje korisnici trebaju za navigaciju i izvršavanje zadataka. UX dizajn se razvio kao rezultat poboljšanja dizajna UI. To je proces poboljšanja zadovoljstva različitih korisnika te se temelji na razumijevanju korisnika i njihovih potreba. Karakterističan pristup UX dizajna je holistički pristup koji uzima u obzir sve aspekte proizvoda (primjerice, za web sjedište bi to bili vizualni dizajn, dizajn sučelja i navigacije, kvaliteta sadržaja/informacija, tehničke performanse) [20].

Ova dva dizajna su dva odvojena, ali i međusobno povezana načina pristupa dizajnu jer nije moguće govoriti o korisničkom iskustvu bez govora o korisničkom sučelju i obrnuto. Loše dizajnirano sučelje loše će utjecati na korisničko iskustvo, kao što će i loše provedeno istraživanje o korisničkom iskustvu dovesti do lošeg dizajna korisničkog sučelja. Obzirom da je UX dizajn širi od UI dizajna, povezanost ovih dvaju područja može se prikazati kao na slici 1. U idućim poglavljima bit će detaljnije objašnjen pojedini pristup dizajnu i njegove karakteristike.



Slika 1. Povezanost UI i UX dizajna [2]

2.1. Dizajn korisničkog sučelja

Dizajn korisničkog sučelja (UI) odnosi se na vizualne elemente digitalnog proizvoda. Fokus je uglavnom na izgled i stil, a ne na cjelokupno iskustvo (poput dizajna korisničkog iskustva). Dizajn korisničkog sučelja ima veliki utjecaj na cjelokupno korisničko iskustvo kao "površina" digitalnog proizvoda [2]. U području informatike korisničko sučelje (UI) je bilo koji sustav koji podržava interakciju ljudi i računala (eng. *human-computer interaction* – HCI). Sučelje ima i hardverske i softverske komponente i postoji u obliku ulaza (input) i izlaza (output). Ulazna komponenta omogućuje korisnicima da kontroliraju sustav, a izlazna komponenta omogućuje sustavu da prikaže rezultate korisničke kontrole [20]. UI je točka interakcije čovjeka i računala koja omogućuje čovjeku da uspješno upravlja računalom ili uređajem s kojim je u interakciji. Mora biti intuitivno, efikasno i prijateljsko (eng. *user-friendly*) tako da interakcija bude laka, bez napora i ugodna korisniku. Dobro korisničko sučelje je funkcionalno, pouzdano i ugodno za korištenje. Dizajn korisničkog sučelja trebao bi minimizirati trud koji korisnik mora uložiti u interakciju s proizvodom i pomoći korisnicima da s lakoćom ostvare svoje ciljeve.

Dizajneri koriste metode dizajna usmjerenog na korisnika (intervjui s korisnicima, izravna zapažanja, itd.) kako bi naučili o svojoj ciljnoj publici i osigurali da im je vizualni jezik koji uvode u korisničko sučelje dobro prilagođen. Važno je učiniti korisničko sučelje estetski ugodnim (a ne samo funkcionalnim) jer takva sučelja krajnji korisnici percipiraju upotrebljivijima [5].

Interaktivnost (eng. *interactivity*) je jedna od ključnih karakteristika multimedijskih sučelja. Interaktivnost je međusobna komunikacija između medija i korisnika potaknuta različitim tehnološkim značajkama (korisnik i mediji međusobno komuniciraju). Kako tehnologije napreduju, korisnička sučelja postaju sve interaktivnija te ona nude korisnicima novi način pristupa i doživljaja medijskog sadržaja.

2.1.1. Što čini dizajn korisničkog sučelja dobrim?

UI dizajn je estetski dizajn svih vizualnih i interaktivnih elemenata sučelja nekog proizvoda. UI dizajneri u kontekstu informatike (softvera) odgovorni su za dizajn web sjedišta, web i mobilnih aplikacija, desktop aplikacija. Prije početka dizajniranja korisničkog sučelja, važno je znati što ih zapravo čini dobrim. Glavni cilj dobrog korisničkog sučelja je da bude intuitivno, interaktivno i jednostavno za korištenje, ali i atraktivno.

Karakteristika "atraktivnog dizajna" je prilično subjektivna, drugim riječima, osim što su intuitivna, interaktivna i jednostavna za korištenje, korisnička sučelja trebaju biti i simpatična i ugodna za korištenje korisnicima. Intuitivnost je jako važan aspekt dobro dizajniranog proizvoda, čineći sučelje ugodnim i jednostavnim za korištenje [2].

UI se odnosi na grafički raspored svih elemenata aplikacije ili web sjedišta kao što su gumbi, tekst, slike, klizači, tekst za unos, animacije prijelaza,… UI dizajneri su većinom grafički dizajneri (danas još uvijek prevladavaju grafička korisnička sučelja - GUI). Osim je lijepo napravljenog dizajna (eng. *delightfulness*), korisniku je važno da brzo obavi zadatke za

koje mu služi web sjedište ili aplikacija, a ta karakteristika naziva se upotrebljivost (eng. *usability*).

Neki od principa kreiranja dobrog UI dizajna su:

- jasnoći dati prednost nad kompleksnošću,
- reducirati kognitivno opterećenje (*cognitive load*),
- UI dizajn treba biti nevidljiv,
- UI treba iskomunicirati brand,
- korisnik treba imati kontrolu.

Dodatno, važan je responzivni UI dizajn te mogućnost personalizacije i digitalna pristupačnost (eng. *accessibility*). Dobro dizajnirano sučelje treba biti što jednostavnije s prikazanim samo korisnim elementima, a svi ostali poznati elementi (npr. gumbi) trebaju biti predvidljivi.

Važno je poštivati "oko" korisnika i obratiti pažnju na ključne elemente:

- tekst (veličina, stil, razmak, poravnanje…)
- boje, kontrast, svjetlina
- održavanje konzistentnosti *branda*
- minimiziranje broja akcija za neki zadatak
- odgovarajući UI uzorci (npr. već ispunjene forme)
- grupiranje kontrola i objekata na koje se odnose
- dobro informiranje korisnika (jasne upute i označavanje)
- osiguravanje korisniku da prirodno razumije svaki idući korak

2.1.2. Upotrebljivost dizajna (eng. *usability*)

Upotrebljivost je mjera koja definira koliko dobro određeni korisnik može koristiti proizvod za postizanje učinkovitog i zadovoljavajućeg definiranog cilja. Upotrebljivost nije povezana samo s dizajnom sučelja, već je uključena i u tehničku razinu cijelog sustava. S ciljem ocjenjivanja upotrebljivosti dizajna, korisnicima se može dodijeliti niz jednostavnih zadataka. Što su zadaci brže i uspješnije riješeni, razina upotrebljivosti je veća. Također, upotrebljivost opisuje kako korisnik može učinkovito komunicirati s proizvodom i koliko lako se proizvodom može upravljati [3].

Uspješno realizirana upotrebljivost proizvoda uvelike ovisi o ljudima koji provode istraživanja - oni saznaju koje su korisnikove potrebe i očekivanja od proizvoda, a na temelju tog znanja mogu donijeti bolje odluke o dizajnu. Kvaliteta upotrebljivosti sastoji se od nekoliko komponenti: mogućnost učenja, sposobnost pamćenja, zadovoljstvo, učinkovitost, prevencije grešaka te dosljednost.

Primjerice, korisnici izrađene aplikacije za vrednovanje digitalnih obrazovnih igara su nastavnici osnovnih škola. Potrebe koje zadovoljava aplikacija su besplatna registracija, pregled i filtriranje igara te preuzimanje i upute za korištenje igre. Nastavnici imaju

mogućnost komentiranja i ocjenjivanja igre kako bi naredni korisnici mogli vidjeti recenzije za pojedinu igru.

2.1.3. Ljepota dizajna (eng. *delightfulness*)

Iako je upotrebljivost prioritet za korisnike, također je vrlo važno učiniti sučelje zabavnim, zanimljivim i lijepim, pri tome vodeći računa da dizajn bude i jednostavan. Ljepota dizajna subjektivna je mjera. Korištenjem aplikacije korisnici mogu imati različite emocionalne reakcije na dizajn aplikacije. Ponekad je izazovno ocijeniti ljepotu dizajna promatrajući reakciju korisnika jer korisnici neće nužno uvijek vokalno izraziti oduševljenje aplikacijom. [4].

Izrađena aplikacija ima jednostavniji dizajn te su korištene komplementarne boje koje se lijepo slažu zajedno. Na slici 2. prikazana je stranica dobrodošlice u aplikaciju za nastavnike. Cilj je privući korisnike da se registriraju i prijave u aplikaciju te nastave korištenje.

Slika 2. Stranica dobrodošlice u aplikaciji za vrednovanje digitalnih obrazovnih igara

2.1.4. Dizajniranje elemenata

Prilikom dizajniranja stranice važno je pravilno dizajnirati i rasporediti elemente te će u ovom poglavlju biti navedeno nekoliko primjera kako to postići. Osnovni elementi dizajna su: boja, linija, vrijednost, prostor, oblik, forma i tekstura. Boja pomaže djelovati na raspoloženje korisnika prilikom korištenja aplikacije te je dobro koristiti palete boja kao smjernicu za kombiniranje boja prilikom izrade stranice. Korištenjem različitih linija stvara se tekstura te se lakše usmjerava korisnika prema određenoj točki na stranici. U dizajniranju element vrijednosti se odnosi na to koliko je boja svijetla ili tamna i može se vizualizirati u gradijentu koji prikazuje niz varijacija na jednoj nijansi. Ispravnim korištenjem prostora može

se pomoći korisnicima da dožive dizajn onako kako ga je dizajner zamislio. Razmak između elemenata je važan jer prenatrpan raspored može zbuniti korisnika. Postoje razni oblici elemenata, od kojih su tri osnovna: organski ili prirodni, geometrijski te apstraktni oblici. Forma se odnosi na način na koji elementi zauzimaju prostor, tj. umjesto stvaranja elemenata kroz trodimenzionalni oblik, dizajneri stvaraju izgled oblika na ravnoj površini. Tekstura je jedan od elemenata dizajna koji se koristi za predstavljanje izgleda objekta. Postoje dvije vrsta tekstura, a to su: taktilna i vizualna tekstura.

Element koji treba imati svako web sjedište ili aplikacija je navigacija. Navigaciju treba smjestiti iznad prijeloma (eng. *above the fold*). Prijelom je mjesto na stranici do kojeg se vidi stranica prije nego se počne skrolirati prema dolje. Navigacija može biti vidljiva ili skrivena te se može nalaziti u podnožju stranice, u zaglavlju ili sa strane. Kao primjer, u nastavku na slici 3. i slici 4. su prikazane navigacije u aplikaciji za vrednovanje digitalnih obrazovnih igara.

Slika 3. Navigacija u zaglavlju aplikacije za vrednovanje digitalnih obrazovnih igara

Slika 4. Navigacija u podnožju aplikacije za vrednovanje digitalnih obrazovnih igara

Dobrom organizacijom sadržaja na stranici korisnik se može lakše i brže snalaziti. Organizacija sadržaja uključuje sav multimedijski sadržaj, a neki od popularnih načina organizacije su: male slike (eng. *thumbnail*), kartice (eng. *cards*), karusel (eng. *carusel*), arhive (eng. *archive*) te paginacija (eng. *pagination*). Na slici 5. prikazan je primjer kartice iz aplikacije, a na slici 6. paginacija za stranice na kojima se nalazi popis igara.

Slika 5. Kartica iz aplikacije za vrednovanje digitalnih obrazovnih igara

Slika 6. Paginacija stranica u aplikaciji za vrednovanje digitalnih obrazovnih igara

Postoji nekoliko svojstava koje svaki UI dizajniran element ima, kao i neke koje može i ne mora imati [2]. Svojstva koja svaki element ima su visina (eng. *height*) i širina (eng. *width*) te x,y vrijednosti za pozicioniranje elementa na stranici izraženi u pt. Elementi mogu biti rotirani (eng. *rotation*) i ispunjeni s različitim tipovima ispuna (eng. *color, gradient, image, image + gradient overlay*). Moguće je dodati obrube oko elementa, a postoje 3 tipa: unutarnji (eng. *inner*), centrirani (eng. *centered*) i vanjski (eng. *outer*). Podešavajući radijus kuteva (eng. *border radius*) moguće je dobiti različite oblike. Kao primjer, nacrtan je gumb iz aplikacije u 3 različita oblika (slika 7.). U aplikaciji je korišten srednji oblik gumba s blago zakrivljenim rubovima iz razloga što se najbolje slagao uz kartične prikaze igara.

Slika 7. Različiti oblici button-a

Svaki element može također imati sjenu (eng. *shadow*). Primjer sjene na kartičnom prikazu igrice može se vidjeti na slici 8.

Slika 8. Primjer sjene na kartičnom prikazu

Za ljude je prirodno da veće stvari percipiraju kao važnije, pogotovo kada se usporede s manjima. Isto je tako i u korisničkom sučelju, primjerice veliki gumb čini se važnijim od malog, veliki tekst se čini važnijim od manjeg i sl. Jake boje, poput plave, crvene ili zelene, mogu lako privući pozornost korisnika, dok će svijetle boje poput bijele, svijetlosive ili krem bolje funkcionirati kao pozadina.

Na taj način su u izrađenoj aplikaciji korištene bijela boja kao pozadina, a nijanse narančaste, svijetloplave i tamnoplave za objekte i detalje (Slika 2. Stranica dobrodošlice u aplikaciji za vrednovanje digitalnih obrazovnih igara).

2.2. Dizajn korisničkog iskustva

Dizajn korisničkog iskustva je proces kreiranja dizajna interakcije između korisnika i proizvoda ili web sjedišta. Dizajniranje korisničkog iskustva je proces kojim se dizajneri koriste za stvaranje proizvoda kako bi se korisnicima pružilo zadovoljavajuće iskustvo. UX dizajn se razvio kao rezultat poboljšanja UI-a. Dizajnerske odluke u UX dizajnu vođene su istraživanjem, analizom podataka i rezultatima testiranja, a ne estetskim preferencijama i mišljenjima. Za razliku od dizajna korisničkog sučelja, koji je usredotočen isključivo na dizajn računalnog sučelja, UX dizajn obuhvaća sve aspekte percipiranog doživljaja korisnika s proizvodom ili web sjedištem, kao što su njegova upotrebljivost, korisnost, poželjnost, percepcija i ukupna izvedba [6].

Izrada dobrog dizajna korisničkog iskustva znači uzeti u obzir mogućnost svake radnje koju će korisnik vjerojatno poduzeti te razumjeti korisnikova očekivanja na svakom koraku procesa. Razbijanjem posla izrade korisničkog iskustva na njegove sastavne elemente,

moguće je bolje razumjeti zadatak u cjelini [1]. Dobro korisničko iskustvo je ono koje udovoljava zahtjevima nekog pojedinog korisnika u specifičnom kontekstu u kojem koristi neki proizvod. UX ima kao cilj poboljšati efikasnost: pomoći ljudima da rade brže i manje griješe pri tome. Ljudi više vole svoj posao ukoliko koriste alate koji su prirodni i jednostavni za korištenje. UX dizajner se ne fokusira samo na kreiranje produkta koji je upotrebljiv nego posvećuje pažnju efikasnosti, užitku, zabavi [20].

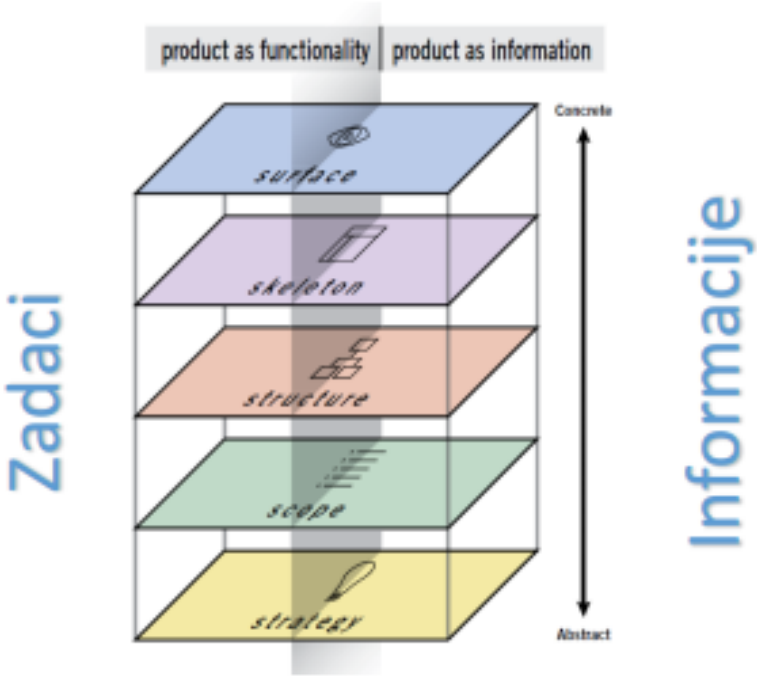
Model elemenata korisničkog iskustva sastoji se od pet ravnina (eng. *planes*):

- ravnina strategije (eng. *strategy plane*)
- ravnina opsega (eng. *scope plane*)
- ravnina strukture (eng. *structure plane*)
- ravnina kostura (eng. *skeleton plane*)
- ravnina površine (eng. *surface plane*)

Ovih pet ravnina čine konceptualni okvir za razbijanje zadatka dizajniranja iskustava na sastavne elemente kako bi se bolje mogao razumjeti problem u cjelini (Slika 9.). Radi se o pristupu odozdo prema gore (eng. *bottom-up*) u kojem svaka ravnina ovisi o onoj ispod, dakle proces nije linearan (npr. kao kod ADDIE modela), što znači da rad na višoj razini počinje dok još nije dovršen na nižoj [20]. Važno je planirati projekt tako da rad na višoj razini ne završi prije rada na nižoj razini.

Prelazeći s niže ravnine na višu, pitanja s kojima se dizajner bavi postaju konkretnije definirana. Na najnižoj ravnini, ne brine se o konačnom obliku krajnjeg proizvoda, dok je na najvišoj ravnini fokusna konkretnim detaljima u izgledu proizvoda [7].

Ukoliko se promatra razvoj web sjedišta, model ravnina može se odnositi i na sadržaj (informacije) i na funkcionalnost (zadatke) web sjedišta. Na slici 11, na lijevoj strani nalaze se elementi specifični za web kao platformu za funkcionalnost, tj. zadaci. Prikazuje se koji su koraci uključeni u proces i kako ih korisnici prolaze da riješe zadatak. Desno su elementi specifični za web kao medij za informacije. Prikazuje se koje informacije se nude i što one znače za korisnike, a korisnici trebaju naći informacije i iskoristiti ih na njima smislen način. Elementi unutar svake ravnine su međusobno povezani bez obzira radi li se o zadacima ili informacijama [21].

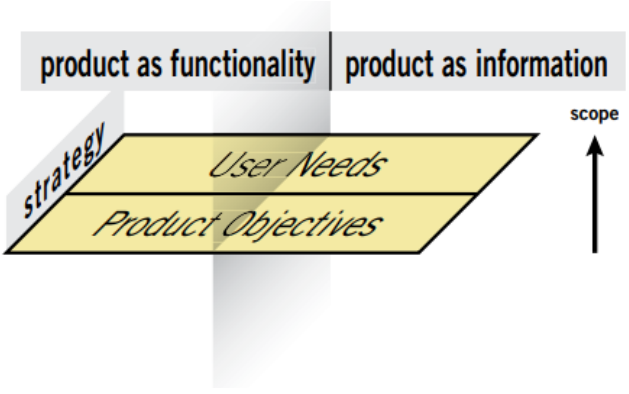


Slika 9. Pet ravnina modela korisničkog iskustva [1]

2.2.1. Ravnina strategije

Strategija (eng. *strategy*) uključuje potrebe korisnika kao i ciljeve proizvoda. Stoga je najbolje vrijeme za početak korištenja metode s pet ravnina tijekom ili nakon faze korisničkog istraživanja u procesu projektiranja. U osnovi, strategija mora odgovoriti na dva pitanja:

1. Što želimo postići proizvodom?
2. Što naši korisnici žele dobiti od proizvoda?



Slika 10. Ravnina strategije [1]

Odgovarajući na prvo pitanje, opisuju se ciljevi proizvoda koji dolaze iz unutarnje organizacije. Drugi naziv za ove unutrašnje strateške ciljeve (eng. *internal strategic objectives*) je poslovni ciljevi (eng. *business goals)* [21].

Drugo pitanje odnosi se na potrebe korisnika, ciljeve nametnute proizvodu izvana [10]. Ciljevi i potrebe korisnika zajedno čine ravninu strategije što je osnova za svaku odluku u procesu dizajniranja korisničkog iskustva te ih treba jasno odrediti. Rezultat ove faze je dokument sa strategijom (eng. *strategy document*).

Kako bi se odgovorilo na navedena pitanja, najprije je potrebno provesti istraživanje kako bi se ustanovili problemi koji se trebaju riješiti zbog korisničkih i poslovnih potreba. Prije nego što se prijeđe na istraživački dio potrebno je potvrditi je li proizvod inovativan, izvediv, poželjan te održiv. Inovativnost predstavlja nešto što još nije stvoreno. Kako bi se odredilo je li proizvod izvediv potrebno je ispitati može li se dizajnirati te postoje li sredstva za izradu poštujući određeno vremensko ograničenje. Proizvod se smatra poželjnim ako je korisnicima važan te ga žele koristiti, a održiv ako se može ažurirati i poboljšavati .

Dionik je svatko iz tvrtke koja je naručitelj projekta tko ima interes ili je uključen u projekt i njegov ishod, poput projektnog tima, izvršnog direktora, voditelja projekta i klijenta. Potrebno je saznati što svi misle i što za njih znači uspjeh projekta, odnosno dobiti informacije o poslovnim ciljevima.
Kako bi se identificirali poslovni ciljevi, dionicima se postavljaju pitanja koja se odnose na to što bi proizvod trebao postići za poslovanje (primjerice, povećati zaradu prodajom više proizvoda).

Korisnicama se postavljaju pitanja o tome što im se sviđa ili ne sviđa u aplikaciji. Korisnici su podijeljeni u dva segmenta: B2B i B2C. Business-to-Business (B2B) je prodaja proizvoda osobama koje rade u drugom poslu ili tvrtkama. Dakle, tvrtke prodaju proizvode drugim tvrtkama. Business-to-Consumer (B2C) je slučaj kad tvrtka prodaje proizvod izravno korisnicima ili potrošačima, poput Amazona.

Može se zaključiti kako se uspješan UX dizajn rađa iz jasne strategije, odnosno da Strategija ima najveći utjecaj na uspjeh ili neuspjeh projekta. Ukoliko je ona dobro osmišljena, korisnici će biti zadovoljni s konačnim proizvodom.

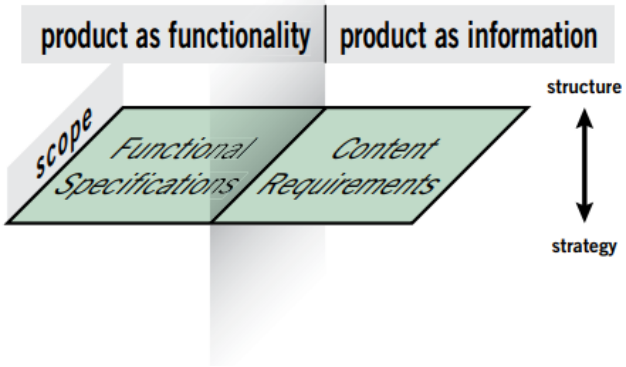
Ravnina strategije za aplikaciju za vrednovanje digitalnih obrazovnih igara

Prije dizajniranja aplikacije za vrednovanje digitalnih obrazovnih igara napravljen je intervju s nastavnicima osnovnih škola. Odgovoreno je na pitanje što želimo postići proizvodom, a to je napraviti intuitivnu aplikaciju koja nastavnicima prikazuje igre namijenjene učenicima nižih razreda osnovne škole kako bi što bolje savladali nastavne sadržaje. Cilj je postići zadovoljstvo nastavnika tako da uspješno pronađu igre i na zabavan način omoguće učenicima da uče i na taj način i oni budu zadovoljni. Naši korisnici, tj. nastavnici, žele dobiti jednostavnu, preglednu i korisnu aplikaciju u kojoj mogu razmijeniti svoja iskustva s ostalim nastavnicima te besplatno pristupiti i neograničeno pregledavati igre. Žele postići zadovoljstvo učenika te povećati njihovu zainteresiranost za određeni predmet, razviti vještine, a sve to na jednostavan i djeci prihvatljiv način.

2.2.2. Ravnina opsega

Opseg (eng. *scope*) je vođen strategijom proizvoda. Unutar ove ravnine istražuje se koje značajke i funkcije unutar opsega, kao i koji elementi sadržaja moraju biti potrebni da bi

se zadovoljile potrebe korisnika. Struktura definira način na koji se različite značajke i funkcije web sjedišta uklapaju, a koje to značajke i funkcije sadrži sjedište definira ravnina opsega. S jasnom vizijom što korisnici žele, moguće je smisliti kako ostvariti strateške ciljeve. Strategija postaje *opseg* kada se potrebe korisnika i ciljevi web sjedišta prevode u specifične zahtjeve za sadržaj (eng. *content requirements*) i funkcionalnosti (eng. *functional specifications*) koji će se ponuditi korisnicima [1].



Slika 11. Ravnina opsega [1]

Pitanja koja se nameću u ovom koraku su: „Zašto se stvara ovo web sjedište? Koja je svrha web sjedišta?“. Funkcionalnost čini skup mogućnosti softverskog proizvoda (zahtjevi koje će korisnik moći izvršiti), dok informacijsku stranu čini sadržaj (multimedijski elementi koje će korisnik moći pregledavati).

Informativna web sjedišta kod kojih prevladava sadržaj obično se izrađuju putem sustava za upravljanje sadržajem (*eng. content managment system - CMS*). Sustav za upravljanje sadržajem može automatizirati potreban tijek rada za dizajniranje i isporuku web sjedišta korisnicima [1].

Definiranje jasnih zahtjeva je važno zbog sljedećih razloga:

1. Treba znati što se radi, tj. jasno popisati ciljeve projekta i kada će biti postignuti te jasno raspodijeliti odgovornosti u timu kako bi se stvorila cjelovita slika, s jasnim vezama između pojedinačnih zahtjeva i s usklađenom terminologijom.

2. Treba znati što se NE radi, a to se odnosi na popis zahtjeva koji su se eventualno kasnije pojavili, a koji se neće uključiti u trenutnu verziju proizvoda.

Do popisa zahtjeva se dolazi istraživanjem potreba korisnika, kao i naručitelja npr. iz neke organizacije/tvrtke. Utvrđuju se potrebe koje korisnici navode, ali i one za koje ne znaju da im trebaju. Ukoliko se radi produkt za tvrtku, uključuju se korisnici iz različitih odjela. Potrebno je voditi računa o zahtjevima hardvera kao što su npr. GPS, kamera i sl. Može se tražiti inspiracija kod konkurencije tj. pregledavati slične proizvode. Neki od savjeta kako napisati specifikaciju zahtjeva su: zapisati zahtjeve kratko i jasno, izbjegavati negaciju (npr. ne navoditi: “Sustav neće dozvoliti korisniku…”), konkretno navoditi zahtjev (umjesto “Najpopularniji članak će biti istaknut…” radije “Članak koji je u prethodnom tjednu imao najviše pregleda, prikazati će sa na vrhu popisa”), koristiti jezik razumljiv svima (npr. ne “Koristit će se RWD...” jer svi ne znaju što kratica RWD znači), koristiti objektivne opise (npr. ne navoditi “Web sjedište će imati moderni dizajn“ jer „moderno“ ne znači isto za

različite korisnike), neke zahtjeve definirati kvalitativno (umjesto “Sustav će biti visokih performansi…” radije “Sustav će podržati barem 1000 korisnika istovremeno…”) [21].

Ravnina opsega za aplikaciju za vrednovanje digitalnih obrazovnih igara

U nastavku će kao primjer biti navedeni funkcionalni zahtjevi te zahtjevi sadržaja aplikacije za vrednovanje digitalnih obrazovnih igara.

Funkcionalni zahtjevi:

1. Aplikacija omogućuje novom korisniku (nastavniku) registraciju.
2. Aplikacija omogućuje postojećem korisniku prijavu u sustav.
3. Aplikacija nakon prijave korisnika omogućuje pregled podataka o digitalnim obrazovnim igrama
4. Aplikacija nakon prijave korisnika omogućuje pregled ocjena i komentara drugih korisnika o digitalnim obrazovnim igrama.
5. Aplikacija nakon prijave korisnika omogućuje ocjenjivanje i komentiranje digitalnih obrazovnih igara
6. Aplikacija nudi nastavniku dodatnu opciju za unos i ažuriranje podataka o digitalnim obrazovnim igrama u bazu.

Zahtjevi sadržaja - popis sadržaja:

1. Podaci o digitalnim obrazovnim igrama: naziv igre, opis igre, preporučena dob učenika, područje (čitanje, pisanje, matematika, vrijeme, novac, glazba, komunikacija, boje, ponašanje), poveznica na preuzimanje (URL), platforma, mala slika (*thumbnail)* igre, poveznica na Youtube video.
2. Navigacijski izbornik koji omogućuje korisniku intuitivni odabir željenih opcija.
3. Upute za korisnika prilikom registracije, prijave, pregleda podataka o igri te ocjenjivanja i komentiranja igara.
4. Poruka o neuspješnoj prijavi postojećeg korisnika
5. Poruka o neuspješnoj registraciji novog korisnika.
6. Poruka o nemogućnosti dohvaćanja podataka o digitalnim obrazovnim igrama iz baze.
7. Poruka korisniku ako za određenu igru ne postoje recenzije.

2.2.3. Ravnina strukture

Struktura definira kako smisleno povezati zahtjeve iz prethodne ravnine opsega, odnosno bavi se definiranjem načina na koji korisnici dolaze do određenih funkcija ili informacija. U ovoj ravnini započinje prijelaz s apstraktnih na konkretnije elemente korisničkog iskustva.

Ukoliko se promatra web sjedište, ravnina kostura definira položaj elemenata u sučelju na nekoj stranici, dok struktura definira kako su korisnici došli do te stranice i gdje mogu dalje otići. U vezi sa strukturom se promatraju dva aspekta:

1. dizajn interakcije (kao dio dizajna UI) – odnosi se na kreiranje iskustva korisnika i opisuje opcije za izvođenje i dovršavanje zadataka (funkcionalni aspekt)

2. arhitektura informacija – strukturiranje UX vezano uz prenošenje informacija korisnicima (organizacija, grupiranje, redanje i prezentacija sadržaja).

Struktura interakcije opisuje moguća ponašanja korisnika i definira kako će se sustav prilagoditi i odgovoriti na ponašanja. Umjesto dizajniranja softvera koji će raditi najbolje za tehnologiju, treba dizajnirati softver koji je najbolji za korisnika. Pri tome se koriste konceptualni modeli kao percepcije korisnika o tome kako će se ponašati interaktivne komponente koje su kreirane (npr. konceptualni model za komponentu kolica/košarice za e-kupovinu na web sjedištu je kontejner ("Dodaj u košaricu"). Navedeni primjer prihvaćen je kao konvencija i zato se uobičajeno koristi. Konceptualne modele treba konzistentno koristiti kroz razvoj dizajna interakcije. Korisnici ih prihvaćaju intuitivno, bez objašnjenja.

Arhitektura informacija važna je za produkte orijentirane na informacije (npr. informativno statičko web sjedište), ali i na produkte orijentirane na funkcionalnost (npr. mobiteli). Strukturiranje sadržaja odnosi se na kategorizaciju sadržaja kako bi se korisnik mogao efikasno kretati kroz njega, što se može prikazati organizacijskim i navigacijskim shemama. Dva su pristupa strukturiranja sadržaja:

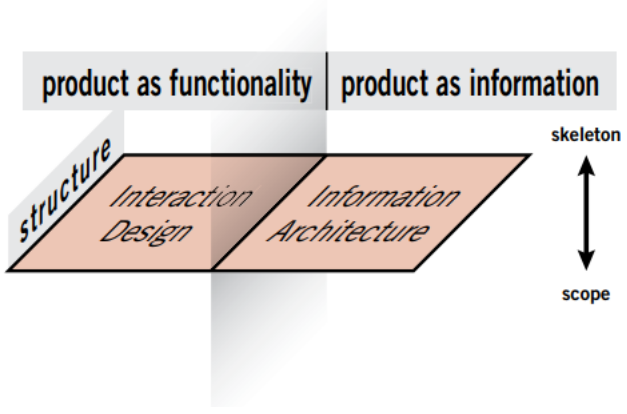
1. pristup *top-down:* glavne kategorije se dijela na podkategorije koje se kasnije "pune" sadržajem
2. pristup *bottom-up*: prvo se analizira sadržaj i zahtjevi funkcionalnosti, grupiranje se vrši iz nižih razina kategorija prema višim.

Broj kategorija (koraka u procesu) nije važan, nego treba biti smislen za korisnika. Struktura web sjedišta treba slijediti na osnovu ciljeva i potreba korisnika. Kod promjena, tj. redizajna strukture pomaže prilagodljiva arhitektura (CMS) [21].

Struktura odgovara na sljedeća pitanja:

1. Kako pomoći korisniku da riješi problem u minimalnom vremenu i najmanjem broju koraka?
2. Razumiju li korisnici arhitekturu projekta?
3. Ispunjava li dizajn očekivanja korisnika?
4. Je li moguće bezbolno promijeniti arhitekturu ako je potrebno?

Glavni zadatak je stvoriti arhitekturu koja ispunjava očekivanja korisnika, a ujedno stvarno pomaže u rješavanju problema korisnika [8].



Slika 12. Ravnina strukture [1]

Najvažniji principi koji čine dobar dizajn interakcije su [11]:

1. Dosljednost

Kada se dizajnira izgled, u korisnikovom umu je potrebno postaviti očekivanja o tome kako bi stvari mogle funkcionirati i kako se kretati po sadržaju, a ona moraju ostati nepromijenjena na svim web stranicama, tj. moraju ostati dosljedna u cijeloj aplikaciji. Iako se sadržaj može promijeniti, osnovne interakcije i proces kroz koji korisnik prolazi trebaju ostati isti i usklađeni s onim što korisnici već znaju.

2. Biti drugačiji samo kada to služi svrsi

Dobro je biti drugačiji kada je promjena napravljena na bolje te ne čini korisnika zbunjenim ili frustriranim.

3. Komponente sličnog ponašanja

Komponente sa sličnim ponašanjem (tj. poruke upozorenja, opisi alata...) trebale bi se ponašati isto na svim zaslonima.

4. Dizajnerski obrasci

Uzorak dizajna je rješenje za višekratnu upotrebu za problem, kao što su gumbi, padajući izbornici, polja za prijavu. Čini radnje i rezultate dosljednim i predvidljivim s onim što ljudi već znaju, jer se koristi obrazac dizajna koji su prije koristili korisnici. Dakle, koristimo ono što korisnici već znaju.

5. Vidljivost

Vidljivost interakcije znači razmišljati o tome što korisnici očekuju od sustava, na primjer, kada korisnik klikne na vezu, on očekuje određeni odgovor, kao što je primjerice preusmjeravanje na ili otvaranje druge stranice.

6. Lakoća učenja

Aplikacija treba biti takva da ju korisnik brzo nauči koristiti te da je jednostavna za uporabu. To znači da će korisnik nakon jednog ili nekoliko korištenja zapamti način korištenja aplikacije.

7. Predvidljivost

Korisnici na bilo kojem ekranu ili stranici trebaju znati gdje se nalaze, kako su tamo stigli, što mogu učiniti, kamo od tuda mogu ići. Ako korisnici mogu odgovoriti na ova pitanja, to znači da je dizajn uspješan jer korisnici mogu točno predvidjeti ishod interakcija.

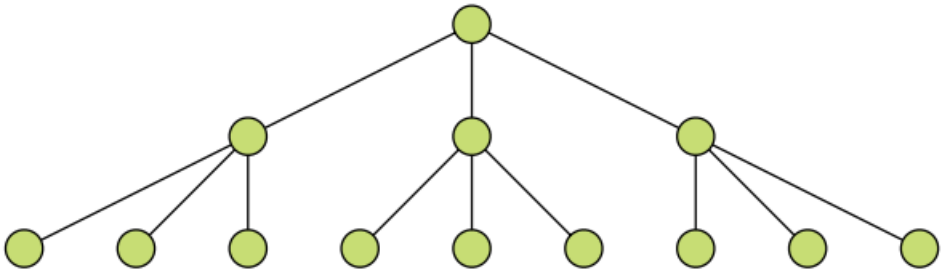
8. Povratne informacije

Povratne informacije trebaju odgovarajućim porukama rješavati moguće pogreške korisnika prilikom uporabe aplikacija. Poruka o pogreškama treba korisniku opisati što se dogodilo, objasniti zašto se to dogodilo i predložiti popravak s ciljem da se spriječiti ponavljanje pogreške.

Postoji nekoliko uzoraka strukture ili načina organiziranja informacija, primjerice hijerarhijska, matrica, prirodna, linearna… Hijerarhijska struktura ili stablo je najčešća

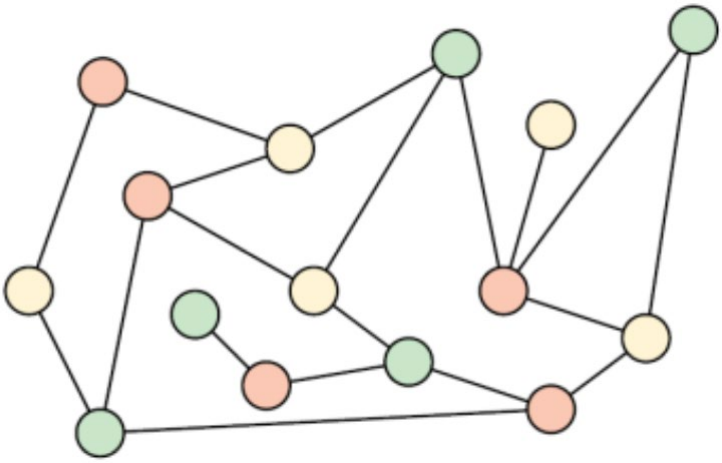
struktura, gdje web-mjesto ima indeksnu stranicu (roditeljski čvor cijele strukture) i niz podstranica (podređeni čvorovi). Podređeni čvorovi predstavljaju uže koncepte unutar općenitije kategorije koju predstavlja roditeljski čvor [11]. Dakle, čvorovi imaju odnose roditelj/dijete. Svaki čvor ne mora imati dijete, ali svaki čvor ima roditelja (Slika 13.).

Linearna struktura je struktura u kojoj se sadržaj nadovezuje jedna na drugi (Slika 14.). Takve strukture se koriste kod manje opsežnih sadržaja kao što su članci. U prirodnoj strukturi čvorovi su međusobno povezani nasumično bez određenih pravila (Slika 15.). Struktura matrica omogućuje korisniku kretanje od čvora do čvora duž dvije ili više dimenzija (Slika 16.).

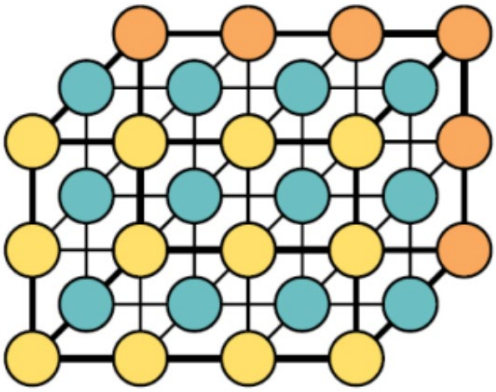


Slika 13. Hijerarhijska struktura [1]

Slika 14. Linearna struktura [1]



Slika 15. Prirodna struktura [1]



Slika 16. Struktura matrica [1]

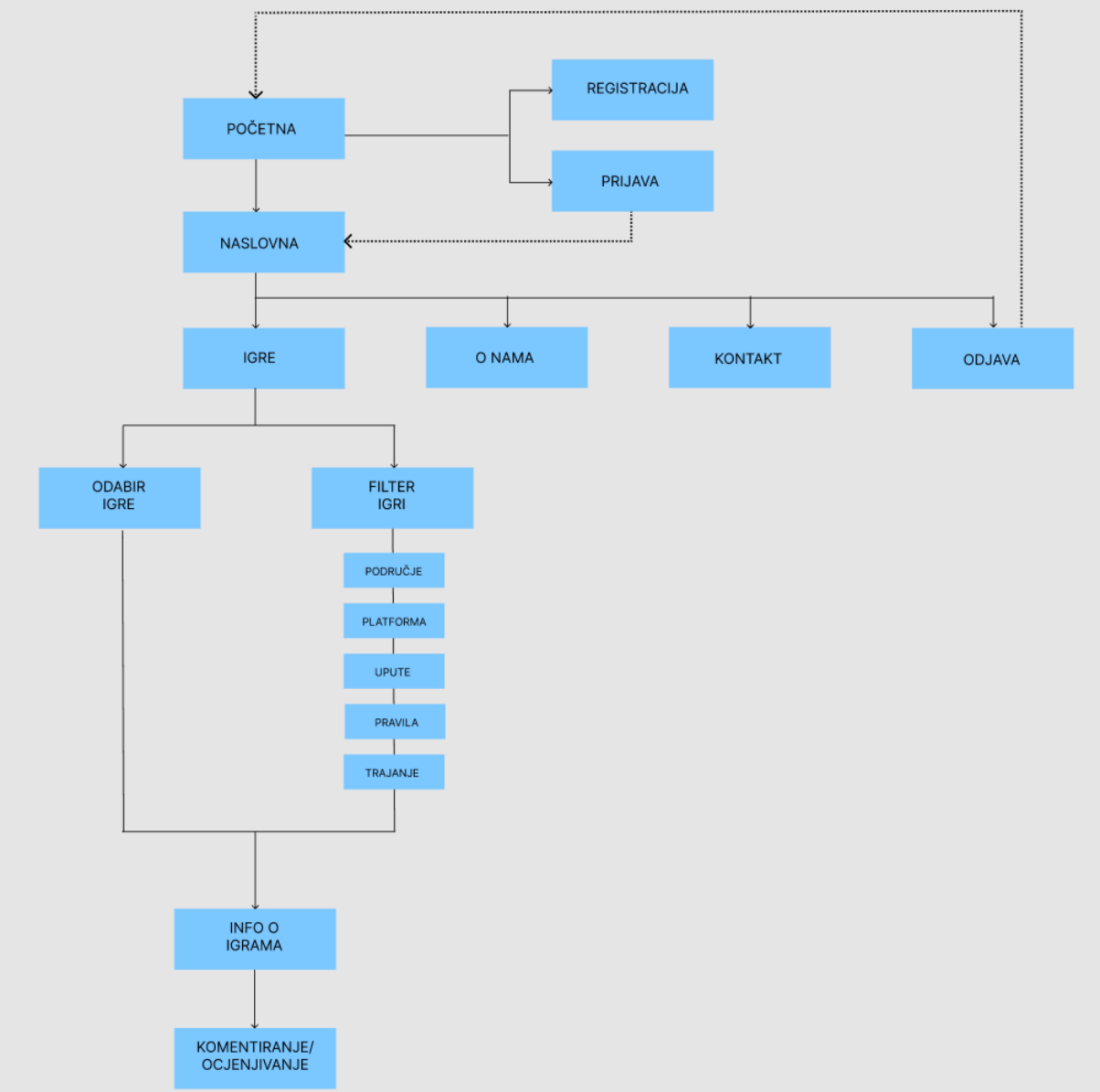
Ravnina strukture za aplikaciju za vrednovanje digitalnih obrazovnih igara

U web aplikaciji za vrednovanje digitalnih obrazovnih igara korištena je kombinacija hijerarhijske strukture i strukture matrica. Primjerice, čvor „Igre“ predstavlja listu svih igara te je on nadređeni čvor, tj. roditelj čvoru „Info o igrama“. Čvorove koji se nalaze ispod roditeljskih nazivamo djecom. Takva povezanost čvorova karakteristika je hijerarhijske strukture. Web sjedište posjeduje filtere za igre čijim korištenjem se omogućava kretanje po aplikaciji karakteristično strukturi matrice. U aplikaciji matrica ima pet dimenzija jer je moguće filtrirati igre prema pet karakteristika, a to su: područje, platforma, upute, pravila i trajanje.

Struktura aplikacije za vrednovanje digitalnih igara prikazana je dijagramom arhitekture za aplikaciju (Slika 17.). Dijagram arhitekture je konceptualni prikaz veza između čvorova te određuje koje kategorije su zajedno grupirane (ne prikazuje navigaciju).

Učitavanjem aplikacije korisnik dolazi na početnu stranicu te ima opciju prijave i registracije. Ako je korisnik već registriran, odabire prijavu, a u protivnom najprije treba obaviti proces registracije. Uspješnom registracijom, korisnik je preusmjeren na prijavu te ima mogućnost korištenja aplikacije.

Na naslovnoj stranici moguće je pročitati više informacija o aplikaciji i igrama. Na stranici “Igre” nalazi se popis igara koje je moguće pregledavati i filtrirati. Može se odmah odabrati željena igra za više informacija ili filtrirati prema: području primjene, platformi za preuzimanje, uputama, pravilima te trajanju. Odabirom nekog ili nekoliko filtera prikazuju se odgovarajuće igre koje zadovoljavaju uvjete. Za svaku igru s popisa može se pročitati naziv, opis te vidjeti slika. Otvaranjem pojedine igre otvara se stranica za tu igru s detaljnijim podacima o njoj ("Info o igrama“). Osim što korisnik može saznati više informacija o igri, može napisati svoj komentar i ocjenu te ga usporediti s mišljenjima ostalih korisnika. Na stranici „O nama“ nalaze se detalji o aplikaciji, dok se na stranici „Kontakt“ nalazi kontaktni obrazac preko kojeg korisnik može poslati poruku administratoru aplikacije. Klikom na gumb “Odjava” korisnika se vraća na stranicu „Početna“.



Slika 17. Dijagram arhitekture

2.2.4. Ravnina kostura

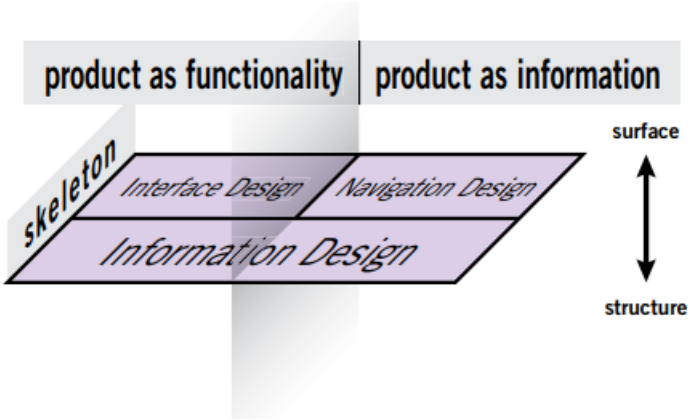
Kostur je konkretan izraz apstraktnije strukture web sjedišta. Ovdje se nastavlja razrađivati isplanirana struktura te se određuju aspekti dizajna sučelja, dizajna navigacije i dizajna informacija. Prethodna ravnina je gledala arhitekturu i interakciju kao cjelinu, a ova se odnosi na individualne komponente i njihovu međusobnu vezu. Ovdje se počinju dizajnirati elementi sučelja — gumbi, tekstualni blokovi, slike i sl., a koji će olakšati korisnikovo razumijevanje i kretanje kroz aplikaciju. Kostur web sjedišta nalazi se ispod ravnine površine te definira poziciju gumba, fotografija i blokova teksta te ostalih elemenata.

Kostur je dizajniran da optimizira raspored elemenata korisničkog sučelja za učinkovitost i jednostavnu upotrebu. Optimiziranjem elemenata na sučelju korisnik će lakše pronaći gumbe koji ga odvode na željenu stranicu, npr. na registraciju, prijavu ili popis igara.

Kod dizajna sučelja potrebno je odabrati prave UI elemente za zadatak koji korisnik treba izvršiti. Raspored tih elemenata na ekranu treba biti takav da budu razumljivi i laki za korištenje. Koji elementi idu na neki ekran određeno je ranije kod strukture, sada se određuje kako su realizirani na ekranu. Dobro dizajnirano sučelje prepoznaje niz akcija koje će korisnik najvjerojatnije napraviti i prikazuje mu elemente sučelja koji su za to najbolji (npr. sustav pamti opcije prethodnog korištenja).

Ciljevi dizajna navigacije su: mogućnost kretanja s jednog mjesta na drugo, prikaz veze između elemenata (struktura veza), veza između navigacije i sadržaja trenutne stranice. Neke često korištene vrste navigacijskih sustava (razine navigacije) su globalna, glavna, lokalna/sekundarna, dodatna i kontekstualna (npr. linkovi unutar teksta na stranici).

Dizajn informacija uključuje različite elemente te predstavlja "ljepilo" koje drži zajedno ostale komponente dizajna. Informacije ne moraju biti samo tekstualne, već ih mogu činiti i slikovni ili videozapis odnosno riječ je o korištenju multimedijskih sadržaja. Sučelje ujedno služi i za prikupljanje informacija od korisnika te daje povratne informacije (npr. poruke o pogreškama) [21].



Slika 18. Ravnina kostura [1]

Wireframes

U ravnini kostura se kreiraju tzv. *wireframes*. To je nacrt koji opisuje sučelje stranice bez puno detalja. *Wireframes* predstavlja "goli" opis svih komponenti stranice i prikaz kako oni zajedno izgledaju. Definira kostur stranica koji se dopunjava u ravnini površine. Izrada *wireframes*-a prvi je korak u izgradnji vizualnog dizajna web sjedišta.

Dizajneri korisničkog iskustva često koriste *wireframes*-e kako bi klijentima, dizajnerima proizvoda i ostalim članovima tima pokazali kako će korisničko sučelje izgledati i funkcionirati. *Wireframes* se koristi kako bi se odgovorilo na ova pitanja:

1. Što će se prikazati u korisničkom sučelju?

2. Gdje će se elementi postaviti na stranicu?

Većina primjera *wireframes*-a uključuje jednostavne linije i okvire s vrlo malo boja ili detalja. Ovi jednostavni oblici predstavljaju UX elemente kao što su izbornici, gumbi, sadržaj i navigacijske funkcije. Na primjer, jednostavan pravokutnik s riječima "Logotip/Početna stranica" može predstavljati mjesto na kojem će biti postavljen logotip tvrtke i označavati da će se logo voditi na početnu stranicu web sjedišta [9].

Neki od ključnih razloga zašto je dobro koristiti *wireframe*-e su [9]:

1. Brzo prenošenje ideja

Wireframes su izvrstan način za brzo prenošenje ideja i dobivanje ranih povratnih informacija koje pomažu kod dizajniranja boljeg proizvoda. U suradnji s klijentima i drugim sudionicima, moguće je surađivati i doći vrlo rano u projektu do konsenzusa o tome kako bi sučelje trebalo izgledati, kako bi trebalo funkcionirati i koje elemente treba uključiti. Dijeljenje *wireframe-a* s klijentom, dizajnerima, razvojnim timom i svima ostalima koji su uključeni u razvoj proizvoda potiče otvoreni dijalog, dobivanje povratne informacije i suradnju.

2. Fokus je na upotrebljivosti

Gledanje *wireframe* dijagrama može dati UX dizajnerima i programerima proizvoda inspiraciju dok analiziraju izgled i usredotočuju se na potencijalnu upotrebljivost konačnog proizvoda. *Wireframe* može ukazati na potencijalne nedostatke. Naglasak na upotrebljivosti u ranoj fazi posebno je važan jer je mnogo jeftinije i lakše riješiti probleme u fazi dizajna nego nakon što se počne izrađivati kod.

3. Štednja vremena i novca

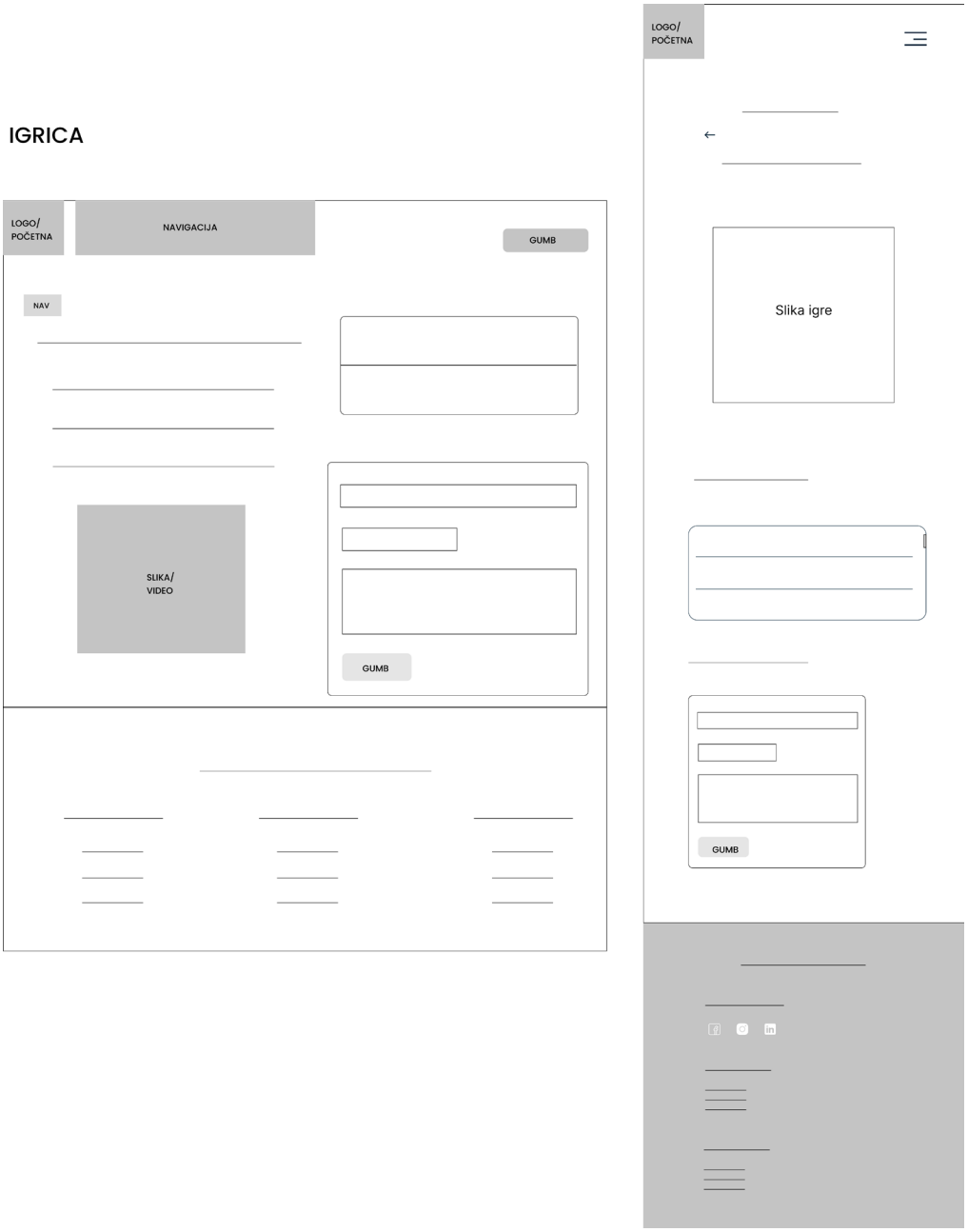
Dobro dizajniran *wireframe* dijagram može uštedjeti vrijeme i novac. Izradom *wireframe*-a može se izbjeći naknadno ispravljanje problema kad je aplikacija već napravljena.

Wireframes **aplikacije za vrednovanje digitalnih igara**

Slijedeći korake dizajniranja korisničkog sučelja, u alatu Figma su izrađeni *wireframe*-i aplikacije za vrednovanje digitalnih obrazovnih igara. Na slikama 19., 20. i 21. prikazani su *wireframe-ovi* u mobilnoj i desktop verziji za karakteristične stranice u aplikaciji: stranica dobrodošlice, stranica igre i stranica informacije o igri.

Slika 19. Wireframe za stranicu dobrodošlice

Slika 20. Wireframe za stranicu igre

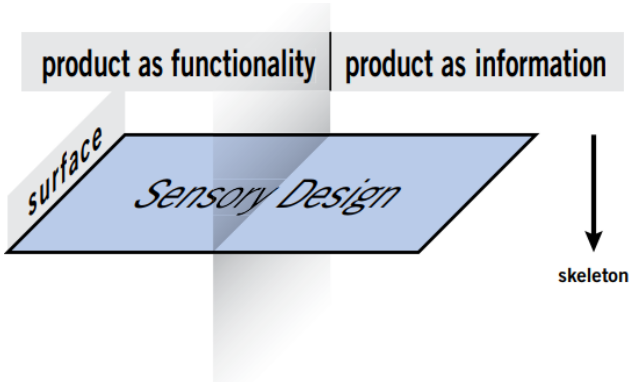


Slika 21. Wireframe za stranicu informacije o igri

2.2.5. Ravnina površine

Ravnina površine nalazi se na vrhu modela elemenata UX, a odnosi se na vizualni dizajn aplikacije odnosno prikazuje se kako bi se elementi na ekranu trebali vizualno prikazati. Sadržaj, funkcionalnost i estetika zajedno ostvaruju gotovi dizajn koji je ugodan za osjetila korisnika, a ujedno zadovoljava sve ciljeve ostalih 4 ravnina ispod ravnine površine. Sav sadržaj i funkcionalnosti spajaju se kako bi se proizveo gotov dizajn. Sve što se stvara u

ovoj ravnini namijenjeno je komunikaciji i poboljšanju razumijevanja svih elemenata na zaslonu, a to se čini odabirom pravog fonta, boja, tekstura,…



Slika 22. Ravnina površine [1]

Grafički ugodno sučelje može dovesti do većeg broja posjeta sjedištu kao rezultat zadovoljstva korisnika. Dizajn je dobro napravljen ukoliko su zadovoljeni navedeni uvjeti: pojašnjava i pomaže korisnicima da razumiju što im je dostupno i kako su elementi povezani, daje naznaku koje radnje korisnik može poduzeti te mu daje jasne informacije [12]. Postoji nekoliko načela vizualnog dizajna kojih se dobro pridržavati prilikom izrade web sjedišta, a to su: poravnavanje, vizualna hijerarhija, ravnoteža ili balans, kontrast, čitljiv tekst, dosljednost i boja [21].

1. Poravnanje

Prilikom prikazivanja teksta na web sjedištu/aplikaciji dobro je obratiti pozornost na prilagođavanje teksta korisniku za lakše čitanje. Bolje je prikazati tekst poravnan na lijevu stranu te da se čita odozgo prema gore jer su korisnici navikli na takvo poravnanje i u ostalim medijima, knjigama i sl. Čitanje centriranog teksta iziskuje više koncentracije i napora za korisnika. Na isti način kao i za tekst treba obratiti pažnju na poravnanje slika, videozapisa i ostalog.

2. Vizualna hijerarhija

Načelo vizualne hijerarhije odnosi se na praćenje različitih elemenata dizajna prema njihovoj važnosti. Ako korisnik ne nailazi na probleme prilikom traženja željenih elemenata na sjedištu, to znači da je vizualna hijerarhija jasno napravljena. Primjerice, kako bi hijerarhija bila jasnija, mogu se koristiti različite veličine slova da bi se korisnicima naznačilo koji su dijelovi sadržaja važni. Također, dobro je označiti različitim bojama važnost elemenata na sjedištu.

3. Ravnoteža ili balans

Načelo ravnoteže odnosi se na zadovoljavajući raspored ili omjer elemenata dizajna. Ravnoteža se javlja kada postoji jednako raspoređena (ne nužno i simetrična) količina vizualnog signala s obje strane zamišljene osi koja prolazi kroz sredinu zaslona. Ta os je često okomita, ali može biti i vodoravna. Pri stvaranju ravnoteže važna je površina koju

zauzima element dizajna, a ne samo broj elemenata. Primjerice, ako se na jednoj strani nalazi jedan veliki element, a na drugoj jedan mali element, dizajn bi djelovao neuravnoteženim.

Ravnoteža na sjedištu može biti:

- Simetrična: elementi su simetrično raspoređeni u odnosu na središnju imaginarnu os.
- Asimetrična: elementi su asimetrično raspoređeni u odnosu na središnju os.
- Radijalna: elementi izlaze iz središnje, zajedničke točke u kružnom smjeru.

4. Kontrast

Kontrastom se prikazuje primjetna razlika (npr. u veličini ili boji) između dva elementa na sjedištu s ciljem naglašavanja njihove različitosti.

5. Čitljiv tekst

Tekst je lako čitljiv kada je razlika između boje pozadine i teksta upadljiva (visok kontrast), tako da nema naprezanja očiju. Tekst nije lako čitati kada nedostaje kontrast između teksta i boje pozadine jer se korisniku teško usredotočiti na tekst ako su boje po vrijednostima bliske.

6. Dosljednost

Dosljednost vizualnog dizajna znači da stil i pristup dizajnu trebaju ostati nepromijenjeni na svim zaslonima web sjedišta. Dobro je koristiti elemente s kojima su korisnici već ranije upoznati tako da će im korištenje biti jednostavnije.

7. Boja

Boja je kritični vizualni element većine korisničkih sučelja, jer pobuđuje emocije. Ona je jedan od najučinkovitijih načina predstavljanja identiteta brenda. Neki su brendovi toliko usko povezani s bojama tako da se pri pomisli na njih odmah vizualizira određena boja. To će stvoriti jači osjećaj identiteta tvrtke u svijesti korisnika. Treba obratiti pažnju koriste li se boje dosljedno na svim zaslonima. Također, može se odrediti pomaže li boja u kretanju kroz informacije i razumijevanju onoga što korisnici vide ili se koristi samo za ukras, a moguće je i korištenje u obje svrhe. Uloga boje je skrenuti pozornost oka na najvažnija područja na ekranu i utjecati na emocionalni odgovor.

U ravnini površine kreiraju se: *moodboard* (ili se dovršava), stilski vodič (eng. *style guide*) i *mockup.*

Moodboard je način prenošenja osjećaja o proizvodu koji će se napraviti. Također, služi kao inspiracija za paletu boja i oblike koji će se koristiti kasnije u dizajniranju sjedišta. UI *moodboard* dobar je način za prikazivanje izgleda i dojma o novom projektu te dizajnu korisničkog sučelja. Stvaranje *moodboard*-a može se izvesti na dva načina: ručno ili digitalno [14]. U današnje vrijeme više se izrađuju digitalni *moodboard*-ovi jer se može brže napraviti kreativniji i precizniji prikaz.

Korištenjem *moodboard*-a suradnja u timu postaje lakša jer on omogućuje da se načela dizajna jasno definiraju. Njegova uporaba olakšava svakome u timu izradu materijala koji su u skladu s dogovorenim dizajnom. Također, *moodboard* štedi vrijeme izrade.

Stilski vodič je dokument koji UX dizajner izrađuje kako bi popisao određene karakteristike vezane za dizajn. Napisane karakteristike ostaju konstantne tijekom izrade cijelog web sjedišta pa se na taj način točno zna koji su standardi za upotrebu boja, tipografije, grafičkih elemenata,…. Primjerice, u stilskom vodiču mogu se navesti: vrijednosti boja koje će se koristiti za izradu sjedišta, font za tekstualni prikaz, ikone, gumbi i sl.

Moodboard i stilski vodič aplikacije za vrednovanje digitalnih igara

Na slici 23. prikazan je moodboard aplikacije za vrednovanje digitalnih igara.

Slika 23. Moodboard

U nastavku je prikazan stilski vodič aplikacije za vrednovanje digitalnih igara.

BOJE:

- # 001E33
- # FF6D33
- # 33CCFF
- # FFFFFF

IKONE:

FONT: ‘Manrope’, sans-serif

'Georgia, Times New Roman, Times, serif'

GUMBI:

Započnite korištenje

Zaigrajte igre

Odjava

Prijava

Registracija

Prijava

Registracija

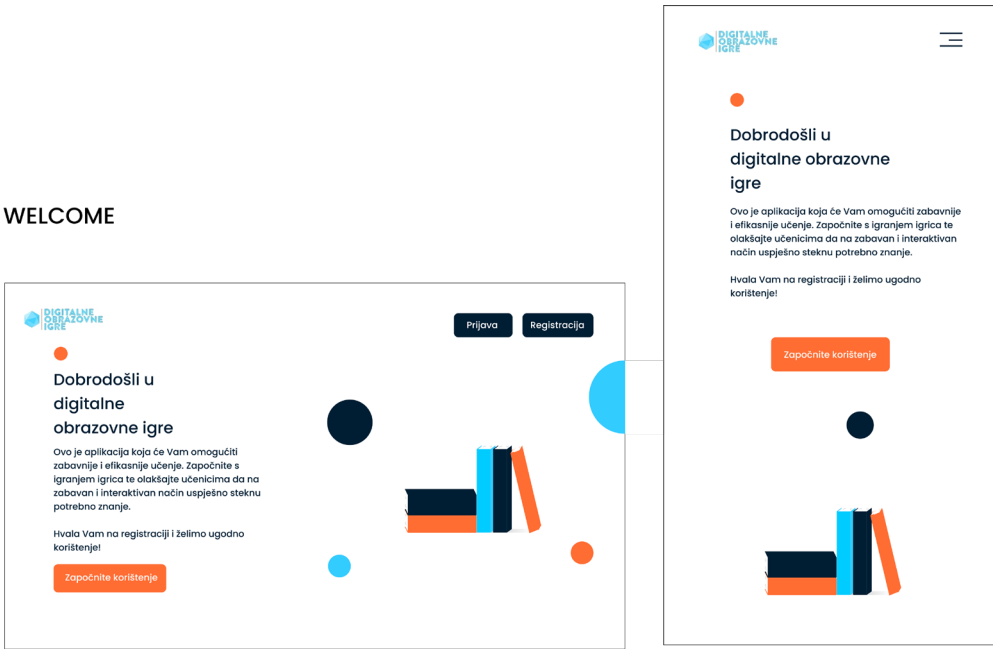
***Mockup* aplikacije za vrednovanje digitalnih igara**

Mockup je vizualni način predstavljanja proizvoda. Dok *wireframe* predstavlja strukturu proizvoda, *mockup* pokazuje kako će proizvod izgledati. Za razliku od *wireframea*, *mockup* je ili srednje ili visoko vjerodostojni prikaz dizajna.

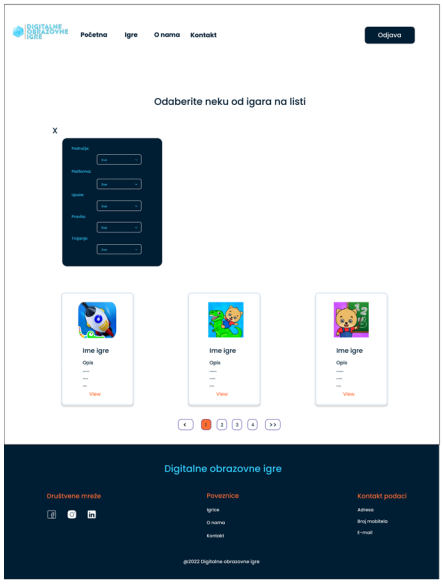
Mockup pomaže donijeti konačne odluke u vezi s shemama boja, vizualnim stilom, tipografijom proizvoda. Uz *mockup* je moguće eksperimentirati s vizualnom stranom proizvoda kako bi se vidjelo što izgleda najbolje. Moguće je zatražiti povratne informacije od

potencijalnih korisnika i odmah napraviti potrebne promjene. Na taj način se uštedi mnogo vremena umjesto da se vraća i prilagođava korisničko sučelje nakon što se pokrene aplikacija/stranica [13].

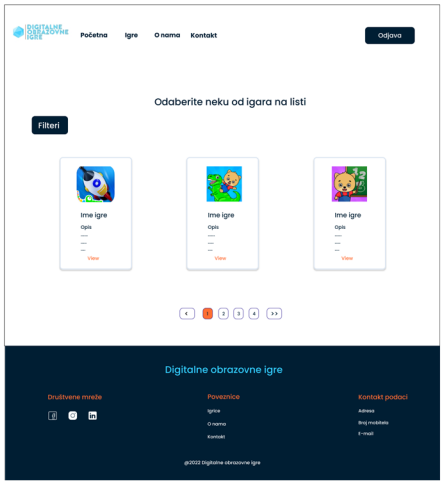
U nastavku bit će prikazani *mockup*-i aplikacije za vrednovanje digitalnih igara izrađeni u Figmi. Na slikama 24., 25. i 26. prikazani su *mockup-ovi* u mobilnoj i desktop verziji za karakteristične stranice u aplikaciji: stranica dobrodošlice, stranica igre i stranica informacije o igri.

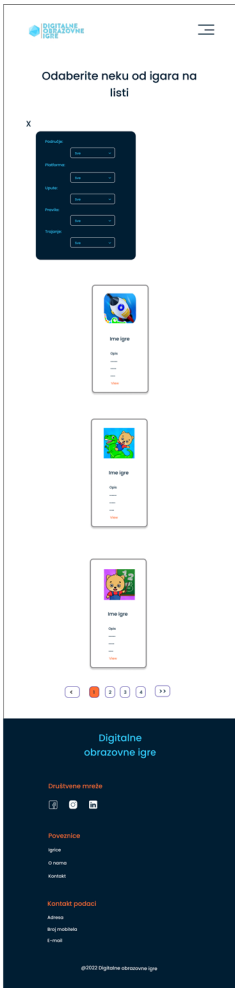


Slika 24. Mockup za stranicu dobrodošlice



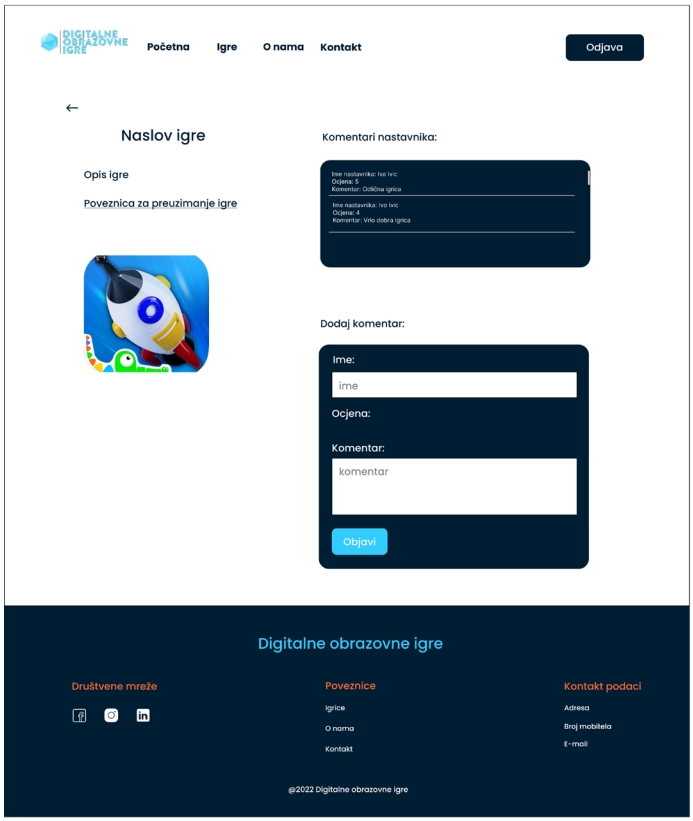
IGRE





Slika 25. Mockup za stranicu igre

IGRE



Slika 26. Mockup za stranicu informacije o igri

2.2.6. Prototip u UX dizajnu

Prototip je simulacija ili ogledna verzija konačnog proizvoda, koju UX timovi koriste za testiranje prije objave aplikacije. Cilj prototipa je testirati i potvrditi ideje prije nego što ih se podijeli s dionicima i na kraju proslijediti konačni dizajn inženjerskim timovima za proces razvoja. Prototipovi su bitni za prepoznavanje i rješavanje problema korisnika sa sudionicima tijekom testiranja upotrebljivosti. Testiranje prototipova s krajnjim korisnicima omogućuje UX timovima vizualizaciju i optimizaciju korisničkog iskustva tijekom procesa dizajna [16]. Prototipovi imaju četiri glavne kvalitete, a to su:

1. Predstavljanje - proizvod je lako predstaviti klijentima pomoću prototipa.
2. Preciznost - vjernost prototipa, što znači njegovu razinu detalja (niska ili visoka vjernost).
3. Interaktivnost - funkcionalnost otvorena korisniku, npr. potpuno funkcionalna, djelomično funkcionalna ili samo za gledanje.
4. Evolucija - životni ciklus prototipa. Neki se izrađuju brzo, testiraju odbacuju i zatim zamjenjuju poboljšanom verzijom (poznato kao "brza izrada prototipova"). Drugi se mogu stvarati i poboljšavati, te se na kraju razviti u konačni proizvod.

Izrada digitalnih prototipova krajnji je dio procesa dizajna. Prototipovi počinju nalikovati konačnom proizvodu što timovima omogućuje testiranje i provjeru valjanosti ideja.

Postoje dvije vrste digitalnih prototipova:

1. Prototipovi niske vjernosti (eng. *low fidelity*): korisnički tijek pomoću *wireframe*-a
2. Prototipovi visoke vjernosti (eng. *high fidelity)*: korisnički tijek pomoću modela

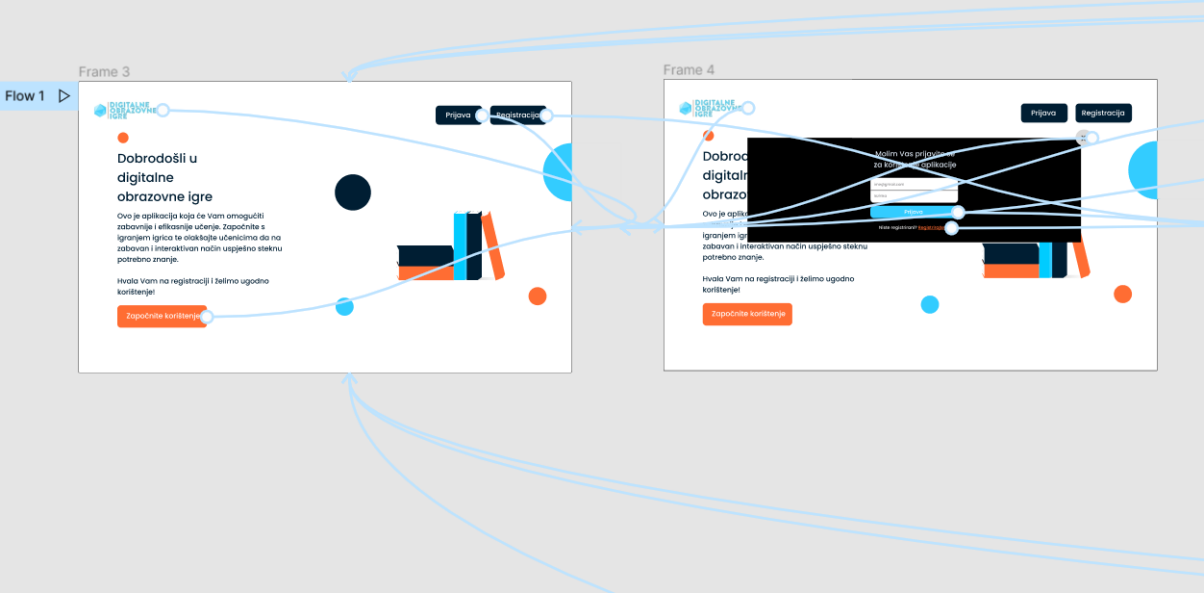
Prototipovi niske vjernosti omogućuju istraživačkim timovima da ocrtaju osnovne korisničke tokove i informacijsku arhitekturu. Prototipovi visoke vjernosti idu u više detalja, testiraju korisnička sučelja, interakcije i način na koji korisnici komuniciraju s proizvodom.

Dizajneri izrađuju prototipove pomoću alata za dizajn kao što su Figma, Adobe XD i drugi. Prednosti izrade digitalnih prototipova su:

1. Realistične interakcije – testiranje s digitalnim prototipovima visoke vjernosti omogućuje UX timovima da vide kako korisnici stupaju u interakciju s konačnim proizvodom, čime se učinkovito izglađuju svi problemi upotrebljivosti.
2. Fleksibilnost – dobro je testirati rano i testirati često. Može se početi s prototipovima koji postaju postupno napredniji kako proces dizajna proizvoda napreduje.
3. Brzina - dok bi papirnati prototipovi mogli biti najbrži način za testiranje ideja, digitalni prototipovi su najbrži način za testiranje problema upotrebljivosti. Jednom kada proizvod dođe do faze razvoja, promjene koštaju znatno više vremena i novca.

Prototip aplikacije za vrednovanje digitalnih igara

Prije kodiranja aplikacije u Figmi je napravljen prototip visoke vjernosti za aplikaciju za vrednovanje digitalnih obrazovnih igara. Na slici 27. prikazan je dio prototipa koji simulira prijavu i registraciju sa stranice dobrodošlice u početnu stranicu aplikacije.



Slika 27. Prototip aplikacije za vrednovanje digitalnih obrazovnih igara

3. Razvoj web aplikacije

Dva pojma koja se često pojavljuju kada je riječ o razvoju aplikacija su *frontend* i *backend*. Oni u osnovi dijele posao programera aplikacija na dva dijela. *Frontend* programer zahtijeva drugačiji skup vještina od *backend* programera. U nastavku bit će objašnjen *frontend* i *backend* razvoj aplikacije za vrednovanje digitalnih igara te kako zajedno funkcioniraju.

3.1. *Backend* aplikacije

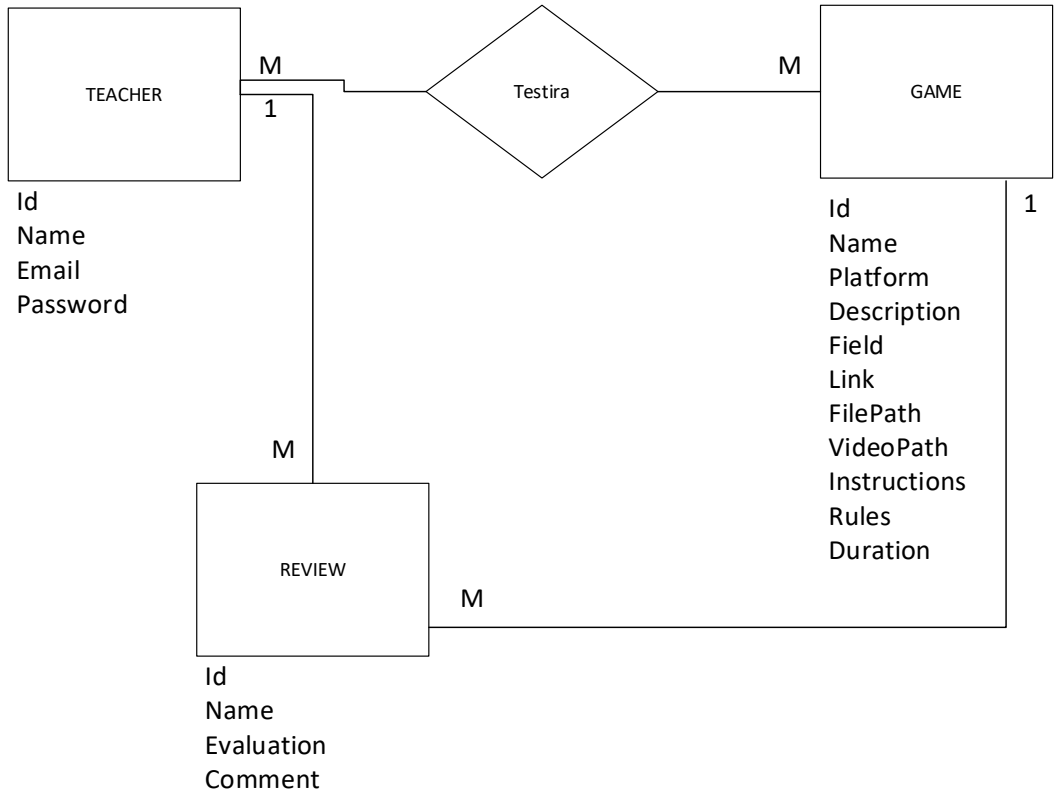
Pozadina (eng. *backend*) aplikacije je također poznata kao poslužiteljska strana web stranice ili aplikacije. Organizira i pohranjuje podatke te osigurava da sve na klijentskoj strani web sjedišta ispravno funkcionira. Iako pozadina ne komunicira izravno s korisnicima, ona igra neizostavnu ulogu iza kulisa dodavanjem ključnih funkcija web sjedištu. Bez čiste i odgovarajuće pozadine, sučelje neće ispravno raditi. Dakle, iako ne postoji izravna interakcija s pozadinom kao s *frontendom*, neizravno ste u kontaktu sa svim procesima koji se događaju na pozadini. Razvoj pozadine uključuje aktivnosti kao što su pisanje API-ja, stvaranje biblioteka i rad sa komponentama sustava bez korisničkih sučelja. Temeljnom funkcijom web-aplikacija obično upravlja pozadina. Na primjer, ako aplikacija ima veliku količinu podataka koji se trebaju prikazati na sjedištu, onda izrada *backend*-a uvelike olakšava dohvaćanje postojećih i dodavanje novih. Također, ako korisnik ima mogućnost za upisivanjem nekih komentara ili izradu vlastitog profila, te podatke je potrebno negdje spremiti. *Backend* dohvaća upisane podatke te ih sprema u bazu kako bi se kasnije mogli prikazati na stranici i koristiti. [17].

Backend aplikacije za vrednovanje digitalnih igara napravljen je korištenjem ASP.NET Core besplatnog web okvira otvorenog koda, kojeg je razvio Microsoft. ASP.NET

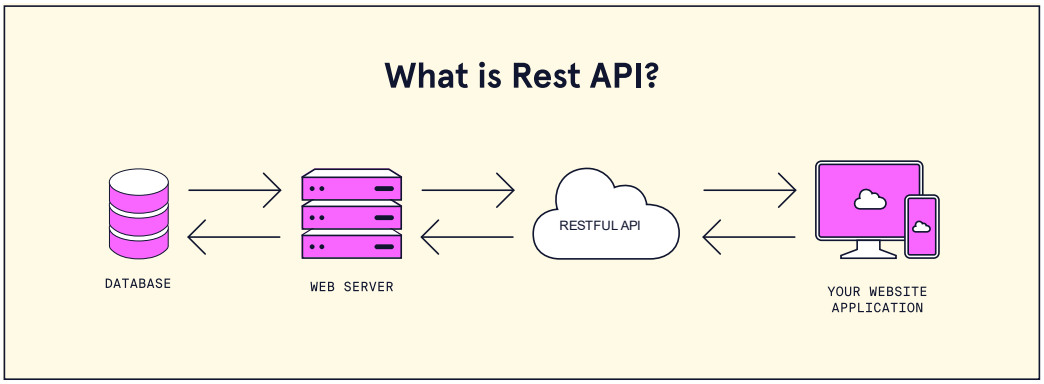
Core je osmišljen kako bi omogućio brzi razvoj API-ja, kompilatora i jezika, a istovremeno pruža stabilnu i podržanu platformu za održavanje aplikacija u radu.

Programski kod izrađene aplikacije organiziran je u slojeve (eng. *layers*) koji olakšavaju izradu i promjene funkcionalnosti unutar aplikacije [18]. Slojevi su podijeljeni u foldere, a to su: Games.Api, Games.Infrastructure te DiplomskiClassLib. Games.Api predstavlja servisni sloj (eng. *service layer*) i u njemu se nalaze REST API kontroleri. Games.Infrastructure je infrastrukturni sloj (eng. *infrastructure layer*) vezan za logiku pristupa podacima. DiplomskiClassLib je sloj domene (eng. *domain layer*) gdje su definirane klase (tablice i njihovi atributi). Za rad s podacima korišten je Entity Framework. Entity Framework je okvir otvorenog koda za .NET aplikacije koji omogućuje rad s podacima koristeći objekte klasa. On kreira model entiteta i veza (eng. *Entity Data Model*) koji je prikazan na slici 28. te ima funkciju stvaranja upita i kreiranja baze podataka [19].

Stvaranjem migracija izrađuju se tablice s najnovijim promjenama te je nakon izrade svake migracije potrebno ažurirati bazu na novo stanje. Nakon stvaranja baze, potrebno je napraviti *controller* datoteke u kojima se pomoću HTTP metoda za REST API dohvaćaju, šalju, mijenjaju te brišu podaci prema potrebi aplikacije. Metode za pristupanje podacima su: POST (za kreiranje novih podataka), GET (za čitanje podataka iz baze), PUT (za ažuriranje ili mijenjanje podataka) te DELETE (za brisanje podataka iz baze). REST API (također poznat kao RESTful API) je programsko sučelje aplikacije (API ili web API) koje omogućuje interakciju s RESTful web uslugama. API je skup definicija i protokola za izgradnju i integraciju aplikacijskog softvera (Slika 29.).



Slika 28. Model entiteta i veza aplikacije za vrednovanje digitalnih obrazovnih igara



Slika 29. REST API, izvor: https://www.codecademy.com/article/what-is-rest

Klasa i objekt osnovni su koncepti objektno orijentiranog programiranja koji se vrte oko entiteta iz stvarnog života. Klasa je korisnički definiran nacrt ili prototip iz kojeg se stvaraju objekti. U osnovi, klasa kombinira polja i metode (funkcija koja definira akcije) u jednu jedinicu. U C#, klase podržavaju polimorfizam, nasljeđivanje i također pružaju koncept izvedenih klasa i osnovnih klasa.

Na primjeru izrađene aplikacije, napravljene su klase "Game" (klasa digitalnih igara), "Teacher" (klasa nastavnika) te "CommentGame" (klasa za komentare igara). Na slici 30. , prikazan je primjer definiranja klase Game koja je deklarirana kao javna (eng.*public*), a njezini atributi su privatnog tipa (eng.*private*). Modifikatori pristupa primjenjuju se na deklaraciju klase, metode, svojstava, polja i drugih članova. Oni definiraju dostupnost klase i njenih članova. Javni (eng.*public*), privatni (eng.*private*), zaštićeni (eng.*protected*) i interni (eng.*internal*) su modifikatori pristupa u C#.

```
public class Game
{
    3 references
    private int _id;
    3 references
    private string _platform;
    3 references
    private string _name;
    3 references
    private string _description;
    3 references
    private string _field;
    3 references
    private string _link;
    3 references
    private string _filePath;
    3 references
    private string _videoPath;
    3 references
    private string _instructions;
    3 references
    private string _rules;
    3 references
    private string _duration;
```

Slika 30. Klasa Game

Klasa Game ima konstruktor s parametrima `int id, string name, string platform, string description, string field, string link, string filePath, string videoPath, string instructions, string rules, string duration` (Slika 31.). Konstruktori su posebne metode u C# koje se automatski pozivaju kada se kreira objekt klase za inicijalizaciju svih članova podataka klase.

```
public Game(int id, string name, string platform, string description, string field, string link, string filePath,
            string videoPath, string instructions, string rules, string duration)
{
    _id = id;
    _name = name;
    _platform = platform;
    _description = description;
    _field = field;
    _link = link;
    _filePath = filePath;
    _videoPath = videoPath;
    _instructions = instructions;
    _rules = rules;
    _duration = duration;
}
```

Slika 31. Konstruktor Game

Svojstva se mogu definirati korištenjem *gettera* i *settera*, kao što je prikazano na primjeru na slici 32. `Get` i `set` su metode koje se koriste za dohvaćanje i postavljanje vrijednosti atributima klase. `GameId` sažima privatno polje. Omogućuje gettere (`get{}`) za dohvaćanje vrijednosti temeljnog polja i settera (`set{}`) za postavljanje vrijednosti temeljnog polja. Primjerice, `_id` je privatno polje kojem se ne može izravno pristupiti. Pristupit će mu se samo preko `GameId`. Stoga `GameId` enkapsulira `_id`.

```
6 references
public int GameId {
    get{
        return _id;
    }
    set{
        _id = value;
    }
}
```

Slika 32. Getteri i setteri

Klasa konteksta (eng. *context*) najvažnija je klasa pri radu s Entity Framework Core-om. Predstavlja sesiju s temeljnom bazom podataka pomoću koje se izvode CRUD (Create, Read, Update, Delete) operacije. Klasa konteksta koristi se za postavljanje upita ili spremanje podataka u bazu podataka. Također se koristi za konfiguriranje klasa domene, mapiranja povezanih s bazom podataka, promjenu postavki praćenja, predmemoriranje, transakcije itd.

Sljedeća klasa GameContext primjer je klase konteksta (Slika 33.). Klasa GameContext je izvedena iz DbContext, što je čini klasom konteksta. Također uključuje skup entiteta za entitete Game, Teacher i CommentGame. OnModelCreating metoda se poziva kada je model za izvedeni kontekst inicijaliziran, ali prije nego što je model zaključan i korišten za inicijalizaciju konteksta. U metodi se definira primarni ključ tablica te vanjski ključevi. Primjerice, tako je definirano da je tablica Game spojena s tablicom CommentGame

na način da jedna igra može imati više komentara, a komentari se generiraju prilikom dodavanja u aplikaciji. Za tablicu Teacher specifično je to što je za atribut Email definirano kako mora biti unique, tj. korisnik se može samo jednom registrirati s istim email-om.

```
public class GamesContext : DbContext
{
    0 references
    public GamesContext(DbContextOptions<GamesContext> options) : base(options)
    {

    }

    5 references
    public DbSet<Game> Games { get; set; }
    3 references
    public DbSet<Teacher> Teachers { get; set; }
    2 references
    public DbSet<CommentGame> CommentGames { get; set; }
    0 references
    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        modelBuilder.Entity<Game>(entityTypeBuilder =>
        {
            entityTypeBuilder.HasKey(g => g.GameId);
            entityTypeBuilder.HasMany(g => g.Comments).WithOne().HasForeignKey(gc => gc.GameId);
        });

        modelBuilder.Entity<Teacher>(entityTypeBuilder =>
        {
            entityTypeBuilder.HasKey(t => t.Id);
            entityTypeBuilder.HasIndex(e => e.Email).IsUnique();
        });

        modelBuilder.Entity<CommentGame>(entityTypeBuilder =>
        {
            entityTypeBuilder.HasKey(c => c.Id);
            entityTypeBuilder.Property(c => c.Id).ValueGeneratedOnAdd();
        });
    }

}
```

Slika 33. Game Context

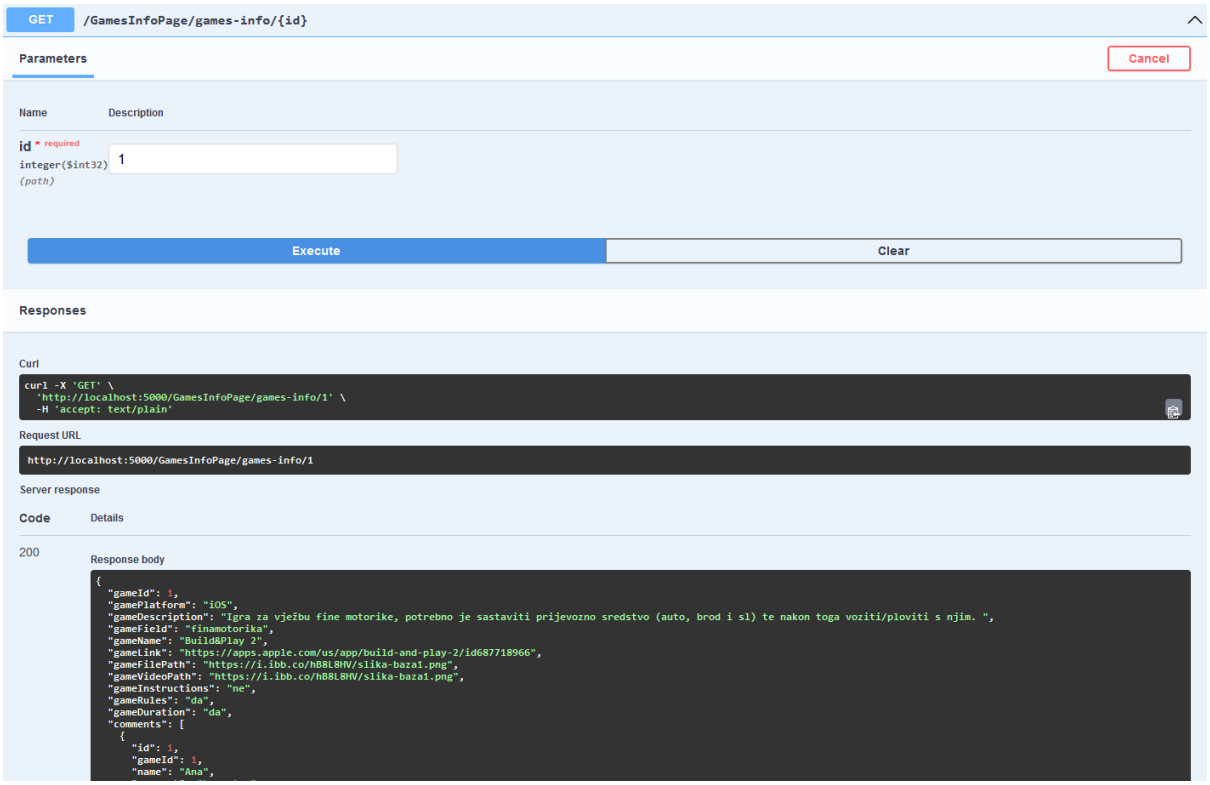
Kontroleri (*eng. controllers*) u MVC aplikaciji logički grupiraju slične vrste radnji. U aplikaciji ASP.NET Core MVC, klasa kontrolera treba i mora biti naslijeđena od osnovne klase kontrolera. Kada klijent (preglednik) pošalje zahtjev poslužitelju, tada taj zahtjev prvo prolazi kroz cjevovod za obradu zahtjeva. Nakon što zahtjev prođe cjevovod za obradu zahtjeva, doći će u kontroler. Unutar kontrolera postoji mnogo metoda (nazvanih akcijskim metodama) koje zapravo obrađuju taj dolazni HTTP zahtjev. Akcijska metoda unutar kontrolera izvršava poslovnu logiku i priprema odgovor koji se šalje natrag klijentu koji je inicijalno podnio zahtjev.

HttpGet, HttpPost, HttpPut i HttpDelete metode su slanja klijentskih podataka ili podataka obrazaca na poslužitelj. HTTP (eng. *HyperText Transfer Protocol*) je protokol koji je dizajniran za slanje i primanje podataka između klijenta i poslužitelja pomoću web stranica. HttpGet, HttpPost, HttpPut i HttpDelete atributi kodiraju parametre zahtjeva kao parove ključeva i vrijednosti u HTTP zahtjevu. Izrađena aplikacija ima nekoliko controller datoteka u kojima se izvode navedeni protokoli. Na slici 34. prikazan je dio koda controllera za dohvaćanje informacija o igricama. HttpGet-om se dohvaćaju svi podaci o odabranoj igri.

Unosom id-a (jedinstvenog identifikacijskog broja igre) izlistavaju se informacije vezane za tu igru (Slika 35.). Na slici su prikazani podaci igre s identifikacijskim brojem 1.

```
[ApiController]
[Route("[controller]")]
0 references
public class GamesInfoPageController : ControllerBase
{
    7 references
    private GamesContext _context;
    0 references
    public GamesInfoPageController(GamesContext gamesContext)
    {
        _context = gamesContext;
    }

    [HttpGet("games-info/{id}")]
    0 references
    public Game GetGame(int id)
    {
        // list of games with name, desc, picture
        return _context.Games.Include(g => g.Comments).Single(b => b.GameId == id);
    }
```

Slika 34. GamesInfoPage kontroler



Slika 35. Swagger kontrolera

HttpPost omogućuje unos novih igara u bazu podataka. Podatke unosi administrator aplikacije, a to je omogućeno instaliranjem Core Admin biblioteke. Administrator ima pristup

svim podatcima o igrama, korisniku (lozinka je šifrirana) i popisima komentara i ocjena svih igara. Za svaku igru unose se podaci: `GameId, GameName, GamePlatform, GameDescription, GameField, GameLink, GameFilePath, GameVideoPath, GameInstructions, GameRules te GameDuration`. Podaci za koje se ne unese niti jedna vrijednost poprimaju vrijednost null, a tu vrijednost mogu imati svi atributi osim `GameId`-a. Nakon potvrde unesenih podataka (funkcija `Submit()`), funkcijom `Add()` spremaju se u bazu. Igra se također može obrisati pomoću HttpDelete-a na način da se upiše `id` igre za koju se žele ukloniti svi podaci iz baze (Slika 36.).

```
[HttpPost("new-game")]
0 references
public ActionResult NewGame([FromBody] Game newGame)
{
    // save new game
    Game app = Game.Submit(
        newGame.GameId,
        newGame.GameName,
        newGame.GamePlatform,
        newGame.GameDescription,
        newGame.GameField,
        newGame.GameLink,
        newGame.GameFilePath,
        newGame.GameVideoPath,
        newGame.GameInstructions,
        newGame.GameRules,
        newGame.GameDuration);

    _context.Games.Add(app);
    _context.SaveChanges();

    return Ok();
}

[HttpDelete("delete")]
0 references
public IActionResult Delete(int GameId)
{
    var Game = _context.Games.Find(GameId);
    _context.Games.Remove(Game);
    _context.SaveChanges();

    return Ok();
}
```

Slika 36. GamesInfoPage kontroler - nastavak

3.2. Frontend aplikacije

Sučelje web ili mobilne aplikacije je dio s kojim korisnik izravno komunicira i obično se naziva *frontendom* aplikacije. Sučelje se sastoji od svega što korisnik vidi prilikom interakcije s web mjestom ili aplikacijom, kao što su boje i stilovi teksta, fotografije, grafikoni i tablice, gumbi, boje, navigacijski izbornik i ostali elementi UI koji su opisani poglavljima o dizajniranju aplikacija.

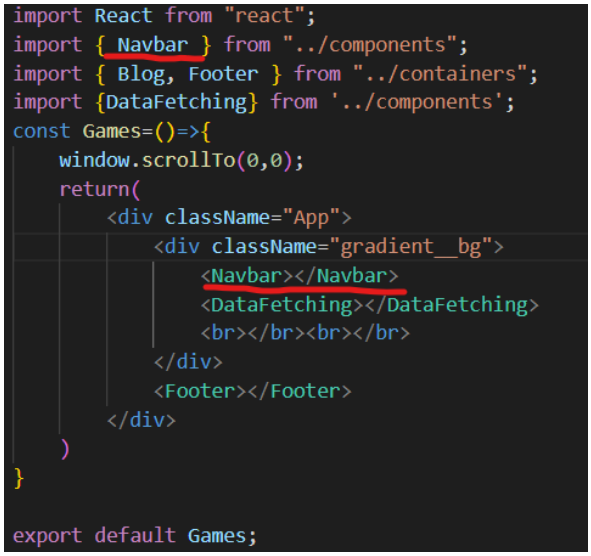
Frontend programeri osiguravaju strukturu, izgled, ponašanje i sadržaj svega što se pojavljuje na zaslonima preglednika kada se otvore web stranice, web aplikacije ili mobilne aplikacije, vodeći računa o pripremljenoj UX/UI dokumentaciji. Programer *frontenda* mora osigurati da web mjesto responzivno reagira, što znači da ispravno radi na uređajima svih veličina. Izvedba aplikacije trebala bi biti stabilna u svakom trenutku, bez obzira na uređaj koji se koristi za pristup aplikaciji [17].

Frontend dio aplikacije za vrednovanje digitalnih obrazovnih igara napravljena je u Javascript programskom jeziku, koristeći ReactJs framework. React JS, koji se obično naziva React, razvojni je okvir korisničkog sučelja, tj. biblioteka za razvoj web stranica i aplikacija. Kreiranje React aplikacije ne odnosi se na *backend* tj. na pozadinsku logiku ili rad s bazama podataka; samo se stvara cjevovod za izgradnju sučelja, tako da ga se može koristiti s bilo kojim *backend*-om. Na primjeru izrađene aplikacije koristio se backend napravljen u .NET-u, a na koji način je povezan s *frontend*-om u React-u bit će objašnjeno u nastavku, uz prikaz nekih najvažnijih dijelova kôda.

Frontend kôd podijeljen je u nekoliko dijelova*; components, containers i pages*. Svrha podjele kôda je ta da se omogući lakše korištenje i implementacija pojedinih dijelova kôda koji se mogu koristiti na više mjesta u aplikaciji. Na taj način, smanjuje se količina kôda i vrijeme pisanja. Primjerice, u *components* mapi napravljena je datoteka navbar.js u kojoj je napisan kod za izgled navigacije (Slika 37.). Taj kod se može koristiti na svim stranicama u aplikaciji na način da se implementira samo jednom linijom kôda (Slika 38.).

```
const Navbar = () => {
  const [toggleMenu, setToggleMenu] = useState(false);
  const logout = async () => {
    await fetch("http://localhost:5000/Auth/logout", {
        method:'POST',
        headers: {'Content-Type': 'application/json'},
        credentials: 'include',
    });
  }

  return (
    <div className ="games__navbar">
      <div className="games__navbar-links">
        <div className="games__navbar-links_logo">
          <NavLink to="/home"><img src={logo2}></img>
            </NavLink>
        </div>
        <div className='games__navbar-links_container'>
          <p><NavLink to="/home" className="listItem" activeClassName="active-link">Početna</NavLink></p>
          <p><NavLink to="/igrice" className="listItem" activeClassName="active-link">Igre</NavLink></p>
          <p><NavLink to="/onama" className="listItem" activeClassName="active-link">O nama</NavLink></p>
          <p><NavLink to="/kontakt" className="listItem" activeClassName="active-link">Kontakt</NavLink></p>
        </div>
      </div>

      <div className='games__navbar-sign'>
        <NavLink to="/"><button type='button' onClick={logout}>Odjava</button></NavLink>
        </div>
```

Slika 37. Navbar kod



Slika 38. Korištenje Navbar tag-a

Statični dio aplikacije moguće je napraviti korištenjem HTML-a i CSS-a, međutim podaci iz baze dohvaćaju se korištenjem određenih React biblioteka i metoda. `UseState hook` je specijalna metoda koja omogućuje dodavanje React stanja komponentama funkcije. Primjerice, na slici 39. je prikazana linija kôda gdje je deklarirana varijabla post i postavljena joj je vrijednost na prazan `array []` te se za ažuriranje varijable poziva `setPost` kako bi joj se promijenila vrijednost.

```
const [post, setPost] = useState([]);
```

Slika 39. UseState hook

UseEffect hook je metoda koja također omogućuje promjene u komponentama, kao što su: dohvaćanje (eng. GET), unos (eng. POST), promjena (eng. PUT) ili brisanje (eng. DELETE) podataka iz baze. To je funkcija koja omogućuje spajanje backend-a aplikacije s frontend-om.

Primjer korištenja useEffect hook-a prikazan je na slici 40. Definirana je funkcija `fetchGames` koja ima ulogu dohvaćanja igara iz baze. Korištena je biblioteka `axios` koja služi kao veza sa http poslužiteljem te se get metodom dohvaćaju podaci sa zadanog url-a. U slučaju da su podaci uspješno pronađeni, u `setPost` će se spremiti dobiveni podaci te će se oni prikazati na stranici, a u suprotnom na stranici se neće ništa pojaviti. U ovom primjeru, funkciji se prosljeđuje prazan niz tako da useState hook ima uvijek početnu vrijednost praznog niza.

Ako se funkciji prosljeđuju varijable, onda se funkcija izvršava samo kad se ta varijabla promijeni. Primjer takvog prosljeđivanja vrijednosti prikazan je na slici 41. gdje se funkcija filtriranja izvršava samo onda kada je korisnik promijenio vrijednost za neku opciju filtriranja. Dakle, ako je korisnik za `selectedCategory` izabrao da želi samo igre vezane za matematiku, početna vrijednost varijable se mijenja te se funkcija izvršava.

```
useEffect(() => {
    const fetchGames = async () => {
        setLoading(true);
        axios.get(`http://localhost:5000/GamesPage/games-page/`)
        .then(res => {
            console.log(res)
            setPost(res.data)
            setFilteredList(res.data)
        })
        .catch(err => {
            console.log(err)
        });
    };

fetchGames();
}, []);
```

Slika 40. UseEffect hook

```
useEffect(() => {
    var filteredData = filterByCategory(post);
    filteredData = filterByPlatform(filteredData);
    filteredData = filterByInstruction(filteredData);
    filteredData = filterByRules(filteredData);
    filteredData = filterByDuration(filteredData);
    setFilteredList(filteredData);
  }, [selectedCategory, selectedPlatform, selectedInstruction, selectedRules, selectedDuration]);
```

Slika 41. UseEffect hook za filtriranje

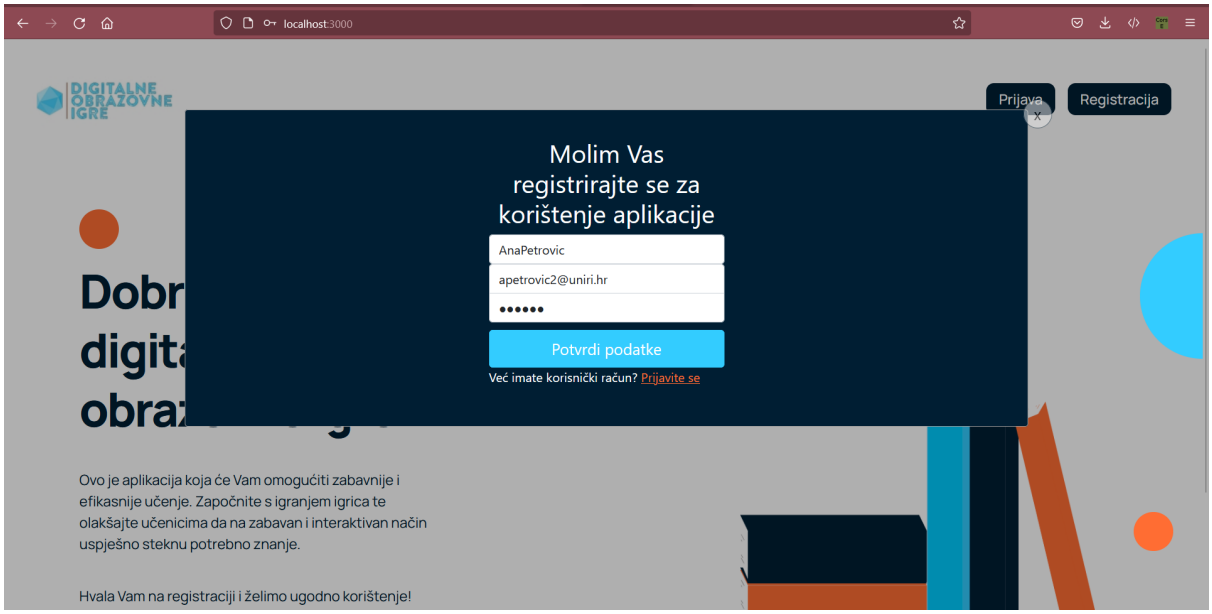
Sljedeći primjer iz aplikacije za vrednovanje digitalnih obrazovnih igara vezan je za komentiranje i ocjenjivanje igre (Slika 42.). Korisnik ima mogućnost ocijeniti kvalitetu igre te dati svoj komentar. To mu je omogućeno korištenjem `useEffect hook-a`, ali s post metodom. Definiran je url koji povezuje funkciju napravljenu u *backend controller*-u za upisivanje komentara. Početne vrijednosti `name, evaluation te comment` su postavljene na prazan string, a `gameId` se prosljeđuje ovisno o igri koja je odabrana. Funkcija `submit` omogućuje unos novog komentara te kad je komentar upisan dohvaćaju se novi podaci za komentare pomoću get metode i prikazuju na stranici pozivom metode `getComment()`.

```
const url="http://localhost:5000/Comment/new-comment"
const [data, setData] = useState({
    gameId : parseInt(id),
    name : "",
    evaluation: "",
    comment: ""
})

function submit(e){
    e.preventDefault();
    axios.post(url,{
        gameId: data.gameId,
        name: data.name,
        evaluation: data.evaluation,
        comment: data.comment
    })
    .then(res =>{
        console.log(res.data)
        const getComment = async () => {
            axios.get(`http://localhost:5000/Comment/games-comment/${id}`)
            .then(res => {
                console.log(res)
                setReviews(res.data)
            }).catch(err => {
                console.log(err)
            });
        };

        getComment();
        })
```

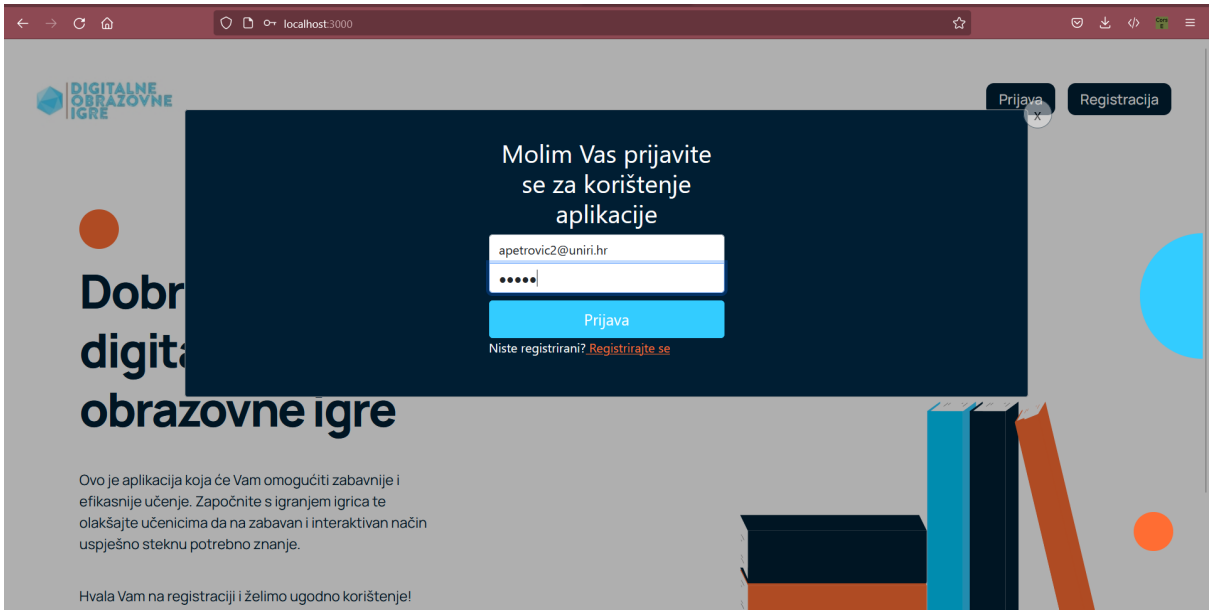
Slika 42. Komentiranje i ocjenjivanje

3.3. Opis korištenja aplikacije

Aplikacija za vrednovanje digitalnih igara besplatna je za korištenje te se svaki korisnik može na jednostavan način registrirati i koristiti ju. Na slici 43. prikazan je način registracije u aplikaciju. Potrebno je upisati korisničko ime, email i lozinku te potvrditi podatke. Ako se korisnik pokuša registrirati s email-om ili korisničkim imenom koji je već iskorišten, onda mu se javlja odgovarajuća poruka. Nakon uspješne registracije, korisnika se preusmjerava na prijavu prikazanu na slici 44. Korisniku se javlja odgovarajuća poruka pogreške u slučaju da upiše pogrešan email, lozinku ili ne upiše jedno ili oba polja.



Slika 43. Registracija u aplikaciju za vrednovanje digitalnih obrazovnih igara



Slika 44. Prijava u aplikaciju za vrednovanje digitalnih obrazovnih igara

Nakon uspješne prijave otvara se naslovna stranica na kojoj je moguće pročitati više informacija o aplikaciji, igrama i sl. (Slike 45.).

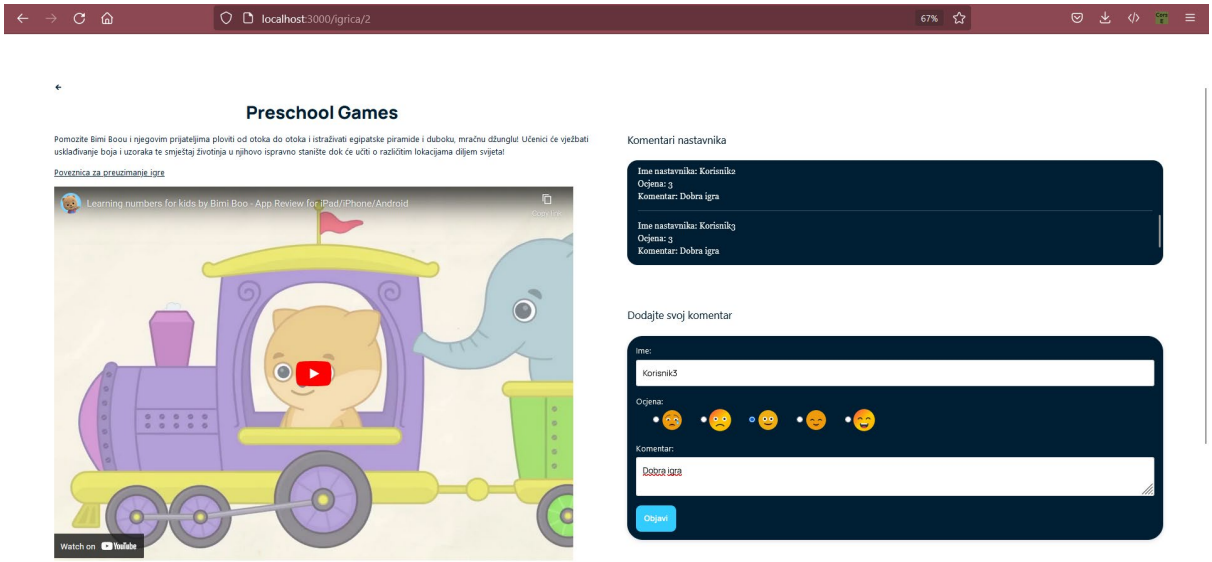
Slika 45. Naslovna stranica aplikacije za vrednovanje digitalnih obrazovnih igara

Klikom na karticu Igre otvara se popis igara za vrednovanje i odabir (Slika 46.). Također korisnik ima i mogućnost filtriranja igrica prema: području primjene igre (fina motorika, matematika, aktivnosti svakodnevnog života, komunikacija, pisanje, vrijeme i boje), platformi za preuzimanje (Android, iOS, Web), uputama (igra ima ili nema upute), pravilima (igra ima ili nema pravila) te trajanju (ima ili nema ograničeno vrijeme trajanja). Izbornik za filtriranje igara prikazan je na slici 47.

Slika 46. Popis igara u aplikaciji za vrednovanje digitalnih obrazovnih igara

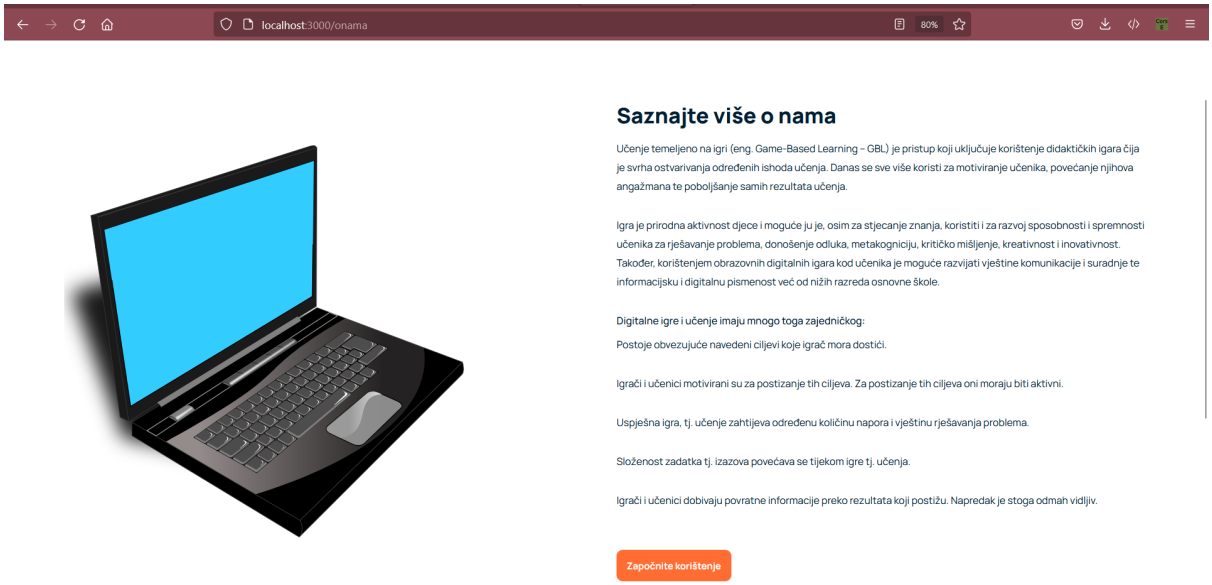
Slika 47. Filtriranje igara u aplikaciji za vrednovanje digitalnih obrazovnih igara

Odabirom igre s popisa otvara se stranica na slici 48. Na njoj su prikazani detalji o igrici, ako igra ima upute u obliku videozapisa prikazan je video, a u slučaju da nema prikazana je samo slika. Korisnik može komentirati igricu, ocijeniti ju te također vidjeti komentare ostalih korisnika.

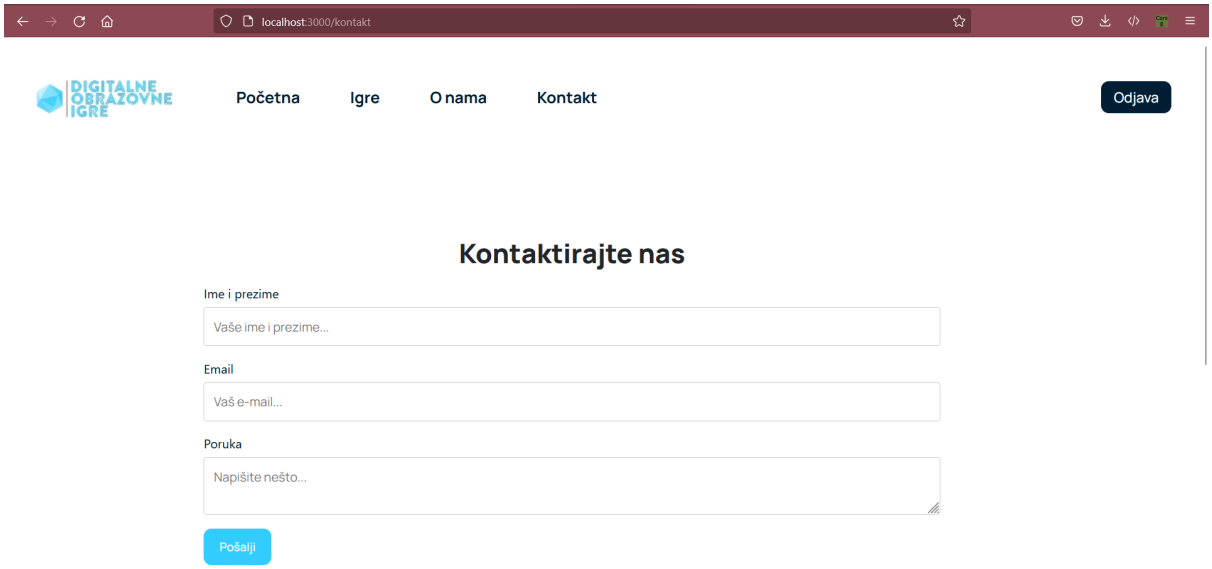


Slika 48. Informacije o igri u aplikaciji za vrednovanje digitalnih obrazovnih igara

Na kartici O nama korisnik može pročitati više informacija o digitalnim obrazovnim igrama te o aplikaciji (Slika 49.), dok se klikom na karticu Kontakt otvara obrazac putem kojeg je moguće poslati email administratoru aplikacije (Slika 50.).



Slika 49. Stranica O nama u aplikaciji za vrednovanje digitalnih obrazovnih igara



Slika 50. Stranica Kontakt u aplikaciji za vrednovanje digitalnih obrazovnih igara

4. Zaključak

Digitalne obrazovne igre imaju nekoliko obrazovnih prednosti. One se mogu koristiti u svrhu razvijanja kognitivnih i motoričkih vještina te pomoći učenicima u korištenju informacijsko komunikacijskih tehnologija. Digitalne se igre temelje na pretpostavci da tijekom igre igrači moraju naučiti, zapamtiti, surađivati i istraživati dodatne informacije kako bi dalje napredovali u igri. Iz tog razloga u ovom radu razvijena je aplikacija za vrednovanje digitalnih obrazovnih igara. Aplikacija nastavnicima omogućuje izabrati i pročitati informacije o igri koju mogu onda i koristiti u radu sa svojim učenicima. Osim informacija o igri aplikacija omogućava i komentiranje zadovoljstva s igrom. Komentari ostalih nastavnika su vidljivi svim korisnicima. Na temelju komentara drugih nastavnika lakše se može zaključiti korisnost igre.

U ovom radu opisan je UX/UI proces dizajniranja aplikacije za vrednovanje digitalnih obrazovnih igara. Dizajn aplikacije je važan jer se njime nastoje ispuniti zahtjevi i omogućiti njeno što ugodnije korištenje. Prilikom dizajniranja važno je paziti na očekivanja korisnika te nastojati pretpostaviti kako će reagirati tijekom korištenja aplikacije. Dakle, prije samog početka izrade dizajna važno je saznati potrebe budućih korisnika aplikacije te osmisliti strategiju. Zatim je potrebno raspisati funkcionalne zahtjeve i zahtjeve sadržaja. Zbog boljeg razumijevanja struktura aplikacije prikazuje se dijagramom arhitekture. U sljedećim koracima planiranja dizajna potrebno je nacrtati izgled aplikacije (*wireframes, mockup, moodboard* i stilski vodič) nakon čega se može krenuti s njenim kodiranje.

Osim dizajna, u radu je opisana i izrada kôda za *backend* i *frontend* aplikacije. Kôd za *backend* je napisan u ASP.NET besplatnom razvojnom okviru koji je dobar izbor za izradu dinamičkih web aplikacija, dok je *frontend* kôdiran u programskom jeziku Javascript koristeći ReactJS razvojni okvir. Koristeći CSS stilski jezik izgled aplikacije prilagođen je *mockup*-u tako da izgleda što sličnije isplaniranom dizajnu.

Svrha izrade aplikacije za vrednovanje digitalnih igara je motivirati nastavnika za korištenje metode učenja putem digitalnih igara u nastavi te zainteresirati učenike za stjecanje novog znanja na zabavan i izazovan način.

Prilikom izrade ovog rada dizajniranje i razvoj aplikacije predstavljali su mi podjednako izazovan zadatak. Po prvi puta sam se susrela s ovakvim načinom planiranja dizajna u kojem je bilo potrebno paziti da se svi koraci UI/UX dizajniranja elemenata naprave ispravno. Pritom sam detaljno planirala dizajn po koracima za što mislim da je vrlo korisno jer se na taj način mogla lakše prenijeti zamisao o izgledu aplikacije u kod. Prilikom razvoja aplikacije ujedno sam i produbila znanje u radu s programskim jezicima i razvojnim okvirima s kojima se do sada nisam detaljno bavila. Za backend sam koristila C# programski jezik i ASP .NET razvojni okvir, dok je *frontend* napisan u JavaScript-u koristeći ReactJS razvojni okvir. Po prvi puta sam radila *backend* za aplikaciju te se susrela s REST API metodama (POST, GET, PUT, DELETE). Smatram da je *backend* bilo izazovnije programirati jer je trebalo uložiti više truda u osmišljanje baze, načina na koji će se ona prikazivati i što će se sve prikazivati. Zadovoljna sam što sam imala mogućnost kroz izradu ovog rada mogla steći nova znanja i iskustva.

5. Literatura

[1] Garrett, Jesse James. *The Elements of User Experience: User-Centered Design for the Web and Beyond, 2nd Edition* (Kindle Edition) Pearson Education, 2011.

[2] eBook by Michael Filipiuk, *"UI Design Priciples",* dostupno na: https://drive.google.com/file/d/1CYlSVrPIKURWkcUts82M1-CP7rxF9sg8/view?usp=sharing

[3] Usability First — Why Usability Design Matters to UI/UX Designers, dostupno na:

https://uxplanet.org/usability-first-why-usability-design-matters-to-ui-ux-designers-9dfb5580116a

[4] A Theory of User Delight: Why Usability Is the Foundation for Delightful Experiences, dostupno na: https://www.nngroup.com/articles/theory-user-delight/

[5] UI Design, dostupno na: https://xd.adobe.com/ideas/process/ui-design/

[6] User experience design, dostupno na: https://en.wikipedia.org/wiki/User_experience_design

[7] UX Design using the Five Planes Method, dostupno na: https://medium.com/designcentered/ux-design-5-planes-method-b1b1d6587c05

[8] 5 Planes of UX/UI: How to Create Convenient User Interfaces, dostupno na:

https://thecustomer.net/5-planes-of-ux-ui-how-to-create-convenient-user-interfaces/

[9] What Is a Wireframe? Why You Should Start Using This UX Design Tool, dostupno na:

https://www.lucidchart.com/blog/what-is-a-wireframe

[10] UX — Strategy (Part 3), dostupno na:

https://medium.com/omarelgabrys-blog/ux-strategy-part-3-6230872b8386

[11] UX — Structure (Part 5), dostupno na:

https://medium.com/omarelgabrys-blog/ux-structure-part-5-4c47c8a54447

[12] UX — Surface (Part 7), dostupno na:

https://medium.com/omarelgabrys-blog/ux-surface-part-7-77986845b49b

[13] Wireframe, Mockup, Prototype: What is What?, dostupno na: https://uxplanet.org/wireframe-mockup-prototype-what-is-what-8cf2966e5a8b#:~:text=A%20mockup%20is%20a%20visual,high%2Dfidelity%20display%20of%20design.

[14] How to Enhance UX Design with Mood Boards, dostupno na:

https://xd.adobe.com/ideas/process/ui-design/how-to-enhance-ux-design-with-mood-boards/

[15] A guide for creating a Style-guide and UI library, dostupno na:

https://uxdesign.cc/making-a-style-guide-%EF%B8%8F-33939edd83ea

[16] What Is a Prototype: A Guide to Functional UX, dostupno na:

https://www.uxpin.com/studio/blog/what-is-a-prototype-a-guide-to-functional-ux/

[17] What Are Frontend and Backend in App Development?, dostupno na:

https://lizard.global/blog/what-are-frontend-and-backend-in-app-development

[18] Layered Architecture with ASP.NET Core, Entity Framework Core and Razor Pages, dostupno na: https://medium.com/aspnetrun/layered-architecture-with-asp-net-core-entity-framework-core-and-razor-pages-53a54c4028e3

[19] What is Entity Framework?, dostupno na:

https://www.entityframeworktutorial.net/what-is-entityframework.aspx

[20] Crux, David. *UI/UX Beginner's Guide* (Kindle Edition), 2021.

[21] Costello, Vic. *Multimedia Foundations, 2nd Edition* (Kindle Edition), Taylor and Francis, 2017.

6. Popis slika

7. Prilozi

Prilog 1: dijagram arhitekture, *wireframes* i *mockup* aplikacije za vrednovanje digitalnih obrazovnih igara

Prilog 2: programski kod za *backend* aplikacije za vrednovanje digitalnih obrazovnih igara

Prilog 3: programski kod za *frontend* aplikacije za vrednovanje digitalnih obrazovnih igara