Dizajn interakcija i razvoj web-aplikacije za zaslone osjetljive na dodir

Cacan, Benjamin

SVEUČILIŠTE U ZAGREBU
GRAFIČKI FAKULTET

BENJAMIN CACAN

DIZAJN INTERAKCIJA I RAZVOJ WEB-APLIKACIJE ZA ZASLONE OSJETLJIVE NA DODIR

DIPLOMSKI RAD

Zagreb, 2022.

SVEUČILIŠTE U ZAGREBU
GRAFIČKI FAKULTET

BENJAMIN CACAN

DIZAJN INTERAKCIJA I RAZVOJ WEB-APLIKACIJE ZA ZASLONE OSJETLJIVE NA DODIR

DIPLOMSKI RAD

Mentor:
doc. dr. sc. Tibor Skala

Student:
Benjamin Cacan

Zagreb, 2022.

Sažetak

U teorijskom dijelu rada objašnjen je dizajn interakcija kao pojam, primjena i najbolja praksa pri oblikovanju interakcija između korisnika i proizvoda koji se koristi te kako on utječe na korisničko iskustvo. U radu su prezentirani principi te najbolja praksa oblikovanja dizajna za kvalitetno korisničko iskustvo i interakciju s računalnim uređajima, fokusirajući se na korisnička sučelja uređaja sa zaslonom na dodir. Nakon dizajna, istražuje se web tehnologija kao popularni način za razvoj aplikacija zajedno s modernim alatima i programskim rješenjima za razvoj internetskih stranica. Uspoređuju se web-aplikacije s izvornim mobilnim aplikacijama i prezentira se koncept progresivnih web aplikacija.

U praktičnom dijelu rada prolazi se kroz proces razvoja interaktivne web-aplikacije za uređaj sa zaslonom velikih dimenzija osjetljivim na dodir. Web-aplikacija razvijena pomoću *React* programske biblioteke, primjer je info-kioska na kojem više korisnika može istovremeno otkrivati sadržaj stupajući u interakciju pomoću različitih gesti dodira. Navode se korištene tehnologije i alati, kako je implementiran dizajn korisničkog sučelja te razvoj i implementacija gesti dodirom kao načina za korisničku interakciju manipulacijom elemenata u korisničkom sučelju.

Ključne riječi: dizajn interakcija, geste dodira, web-aplikacija, *React*

Abstract

In the theoretical part of the paper, interaction design is explained as a concept, application and best practice in designing interactions between the user and the product being used and how it affects the user experience. The paper presents the principles and best practices of designing for quality user experience and interaction with computer devices, focusing on user interfaces of touchscreen devices. After the design, the paper deals with web technology as a popular way to develop applications and touches on modern tools and technologies as solutions for developing websites. It compares web applications with native mobile applications and presents the concept of progressive web applications.

In the practical part, the work goes through the process of developing an interactive web application for a device with a large touchscreen display. The web application developed using the React library is an example of an info-kiosk where multiple users can simultaneously discover content by interacting with different touch gestures. The paper shows the technologies and tools used, how the design of the user interface was implemented, and the development and implementation of touch gestures as a way for user interaction by manipulating elements in the user interface.

Keywords: interaction design, touch gestures, web application, React

SADRŽAJ

1. UVOD

Danas je tehnologija neizostavni alat i velik dio ljudskog života. Ljudi svakodnevno stupaju u interakciju s različitim uređajima od kojih su daleko najviše korišteni pametni telefoni (zasloni osjetljivi na dodir). Uređaji imaju svoje aplikacije od kojih svaka, ovisno o uređaju na kojem se nalazi (platformi), ima svoje zahtjeve, mogućnosti i načine korištenja. Oni koji mogu biti više ili manje usklađeni sa zahtjevima i očekivanjima ljudi odnosno korisnika te aplikacije s kojom stupaju u interakciju. A korisnici će se u većini slučajeva pokušati služiti nekom aplikacijom oslanjajući se i koristeći iskustva i znanja stečena od ranije.

Smatrajući interakciju čovjeka s računalom kao komunikaciju između dvije strane, očito je da se obje strane trebaju služiti istim principima komunikacije kako bi ona bila uspješna. Zato je dizajn interakcija danas sveprisutan proces pri izradih svih vrsta proizvoda (aplikacija) s kojima čovjek kao korisnik treba stupati u interakciju. S računalima se najčešće komunicira razmjenjujući informacije vizualnim putem kroz grafička sučelja, stoga je velika odgovornost na sučeljima koja mogu direktno utjecati na korisničko iskustvo, a time i na uspjeh tog proizvoda.

Istražuju se odgovori na pitanja kao što su: Kako objasniti i oblikovati interakciju?, Kako interakcija utječe na korisničko iskustvo?, Kako dizajnirati za kvalitetno korisničko iskustvo?, Kako na dizajn interakcija utječu različiti uređaji?, Koja je uloga web platforme i po čemu se web-aplikacije razlikuju od mobilnih?

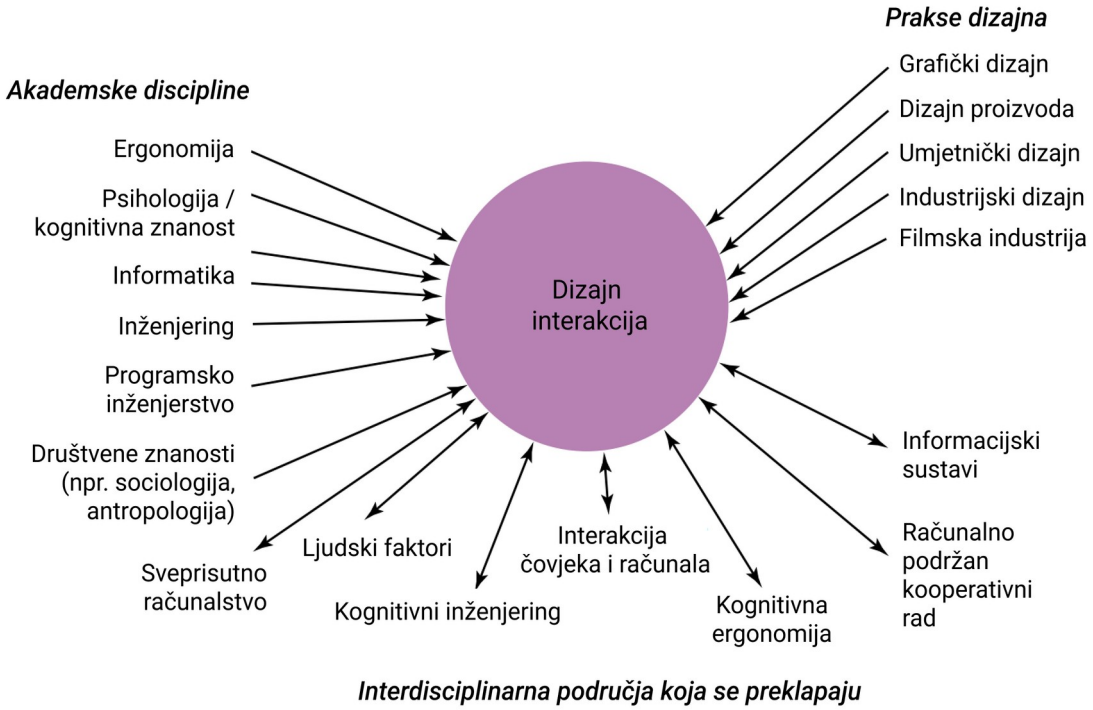
Praktični dio rada prikazat će postupak implementacije dizajna sučelja i razvoja interaktivne web-aplikacije pomoću *React* programske biblioteke. Web-aplikacija će biti razvijena i prilagođena za uređaj sa zaslonom osjetljivim na dodir. S obzirom na površinu zaslona uređaja kojem je dijagonala 140 centimetara, više korisnika istovremeno ima mogućnost biti uz uređaj tako da je zadatak web-aplikacije podržavati višestruku istovremenu interakciju putem gesti dodira.

2. TEORIJSKI DIO

2.1. Dizajn interakcija

Dizajn interakcija (eng. *interaction design*) se u najširem smislu, može se definirati kao oblikovanje (dizajniranje) proizvoda ili usluga na takav način da uspješno služe ljudima s obzirom na ljudsku komunikaciju i interakciju u svakodnevnom životu. Ostale varijante i pokušaji definiranja dizajna interakcija su: “stvaranje korisničkog iskustva koje poboljšava način na koji ljudi rade i komuniciraju”, “oblikovanje prostora za ljudsku komunikaciju i interakciju”, “zašto i kako svakodnevno stupamo u interakciju s računalima”, te “umijeće olakšavanja interakcije među ljudima kroz proizvode i usluge” [1].

Mnogobrojni se pojmovi koriste kako bi se naglasili različiti aspekti onoga što se dizajnira, na primjer: dizajn korisničkog sučelja (eng. UI - *user interface*), dizajn softvera, dizajn usmjeren na korisnika (eng. UCD - *user-centered design*), dizajn proizvoda, web dizajn, dizajn korisničkog iskustva (eng. UX - *user experience design*), i tako dalje… Dizajn interakcija se općenito koristi kao sveobuhvatni pojam za opisivanje navedenih područja, uključujući njihove metode, teorije i pristupe u radu (Slika 1). Temelj je mnogim disciplinama i područjima koje se bave istraživanjem i dizajniranjem računalnih sustava za ljude. Glavna razlika između dizajna interakcija i ostalih pristupa koji se pojavljuju u interdisciplinarnim područjima svodi se na to kojim se metodama i filozofijama služi u proučavanju, analizi i oblikovanju proizvoda, a druga razlika je opseg problema kojim se bave [1].



Slika 1. Odnos između akademskih disciplina, dizajnerskih praksi i interdisciplinarnih područja koja se bave dizajnom interakcije - dvostrane strelice predstavljaju preklapanje (izvor: [1])

Nekad se dizajn interakcija smatrao širokim pojmom, baveći se teorijom i praksom dizajniranja korisničkog iskustva za sve vrste tehnologija i proizvoda, dok je „interakcija čovjeka i računala" (eng. HCI - *human-computer interaction*) imala uži fokus na dizajn i upotrebljivost (eng. *usability*) računalnih sustava. Danas HCI ima vrlo prošireni opseg, puno se više preklapa s generalnim pojmom dizajna interakcija jer HCI uključuje i dizajn interakcija korisničkih sučelja.

Danas se mnoga područja služe dizajnom interakcija, a kako tehnologija postaje neizostavni dio ljudskog života, sve je više ljudi uključeno u poslove koji se bave dizajnom interakcija. Kako bi se mogla stvoriti učinkovita i privlačna korisnička iskustva, dizajneri moraju znati puno o korisnicima i tehnologiji te njihovoj međusobnoj interakciji, emocijama, estetici i poželjnosti. Iako to često nije slučaj, idealno je da se dizajnom interakcija (u praksi) bave

multidisciplinarna područja kako bi se iskoristila različita znanja i vještine. Primjer tih područja mogu biti zanimanja kao što su grafički dizajneri i ostali inženjeri, psiholozi, sociolozi, marketinški stručnjaci i tako dalje [1]...

Danas je dizajn interakcija sveprisutan jer se pokazao nužan za razvoj uspješnih interaktivnih proizvoda. Pojam „interaktivni proizvod" se može generalno koristiti za referiranje na sve vrte interaktivnih sustava, alata, aplikacija, usluga i uređaja. Računalna industrija je relativno brzo shvatila, primarno s internetskim stranicama, kako dizajn interakcija može biti ključan za uspjeh ili neuspjeh interaktivnog proizvoda. No osim računalne industrije, vrijednost interaktivnog dizajna i dobrog korisničkog iskustva prepoznali su i drugi sektori, na primjer pružatelji financijskih usluga, maloprodaje i javi sektor. Cilj je ponuditi proizvod koji je efektivan, privlačan i lagan za korištenje. U području s mnogo konkurencije, kao što je web, internetske stranice su proizvodi koji se moraju uspjeti istaknuti kako bi ih se primijetilo. A s marketinške strane, dobra upotrebljivost (eng. *usability*) internetske stranice uvelike utječe na doživljaj brenda, stopu povrataka i zadovoljstvo korisnika [1].

2.1.1. Korisničko iskustvo – UX

Korisničko iskustvo je pojam koji se referira na to kako se interaktivni proizvod ponaša dok se korisnik njime služe te kako se korisnici uopće njime služe. To jest, svi aspekti ljudskih doživljaja pri interakciji s nečim: jesu li korisnici zadovoljni te kako se osjećaju dok se služe proizvodom [1]. Interaktivni proizvod je opći pojam, u računalnoj industriji su to najčešće internetske stranice ili aplikacije, a korisnici su ljudi koji se koriste tim proizvodima u stvarnom svijetu.

„Svaki proizvod kojim se netko služi - ima svoje korisničko iskustvo: novine, boce kečapa, fotelje, vuneni džemper." — Jesse Garrett (2010.)

John McCarthy i Peter Wright (2004.), pokušavajući definirati pojam iskustva zaključuju kako je vrlo teško to definirati jer koncept istovremeno nejasan, a sveprisutan ljudima (kao što je ribi plivanje u vodi). No pokušali su objasniti esenciju korisničkog iskustva opisujući ga holističkim i metaforičkim terminima koji sadrže balans senzualnih, cerebralnih i emocionalnih doživljaja. Koncept korisničkog iskustva koji je dao Marc Hassenzahl (2010.) sastoji se od samo dva aspekta: pragmatički i hedonistički. Pragmatički jer je bitno koliko je jednostavno, praktično i očito korisniku uspješno se služiti nečim. A hedonistički zbog toga koliko je nešto korisniku izazovno i poticajno za interakciju [1].

Što se tiče „izrade" korisničkog iskustva, to se postiže utjecanjem na razne aspekte korisničkog iskustva pomoću poznatih obrazaca kako bi konačni rezultat bio kvalitetno korisničko iskustvo. Tako da se može reći kako nije moguće dizajnirati korisničko iskustvo samo po sebi, već se dizajnira za određeno korisničko iskustvo. Kao što se ne može stvoriti zadovoljstvo nego samo dizajnirati uvjete koji će dovesti do zadovoljstva korisnika. Ne postoji "formula" s kojom dizajneri interakcija mogu automatski proizvesti kvalitetno korisničko iskustvo. No postoje brojni konceptualni okviri, isprobane metode dizajniranja, smjernice i relevantni rezultati istraživanja.

Mnogo je aspekata korisničkog iskustva i načina na koji im se može pristupiti pri oblikovanju interaktivnog proizvoda. Ključni dijelovi su: upotrebljivost, funkcionalnost, estetika, sadržaj, izgled i dojam, emocionalna privlačnost. Kao što sam naziv govori, korisničko iskustvo „posjeduje" korisnik interaktivnog proizvoda. Zbog toga je u procesu dizajniranja interakcija iznimno bitno ne zaboraviti tko je krajnji korisnik proizvoda, to jest znati iz kojih se točno razloga donose određene odluke pri razvoju proizvoda.

2.1.1.1. Korisnici

Kako dizajneri mogu znati kako oblikovati interaktivne proizvode koji će pružati kvalitetno korisničko iskustvo, to jest odgovarati potrebama korisnika? Dizajnerima je temelj razumijevanje korisnika u kontekstu u kojem oni žive i rade. A s obzirom na to da se potrebe korisnika razlikuju kao što se razlikuju i ljudi generalno, dizajneri moraju razumjeti individualne razlike. Jedno jedinstveno rješenje ne može odgovarati svim grupama ili vrstama korisnika, već korisnici trebaju rješenje koje će odgovarati njihovim potrebama [1]. Što znači da dizajnersko rješenje treba biti oblikovano upravo prema karakteristikama (grupe) korisnika. Kao što su i svakodnevni proizvodi oblikovani za različite vrste korisnika (na primjer prema dobi korisnika), isto se primjenjuje i na interaktivne proizvode.

Za oblikovanje korisničkog sučelja nekog digitalnog proizvoda, dizajneri trebaju moći zamisliti komunikaciju između korisnika (osobe) i aplikacije (softvera). Za potrebno razumijevanje korisnikovih očekivanja postavljaju se pitanja kao što su: Koji su motivi i namjere korisnika?, Koji narativ riječi, ikona i gesti korisnik očekuje?, Na koji način korisniku postaviti očekivanja kako se ne bi razočarao?, Kako korisnik i računalo uspostavljaju smislenu međusobnu komunikaciju? [2].

2.1.1.2. Ciljevi korisničkog iskustva i upotrebljivost

Dio procesa razumijevanja korisnika jest imati jasnu predodžbu o primarnom cilju razvoja interaktivnog proizvoda (za te korisnike). Za lakšu identifikaciju ciljeva, predlaže se klasifikacija s terminima „upotrebljivost" (eng. *usability*) i „ciljevi korisničkog iskustva" (eng. *user experience goals*) [1].

Upotrebljivost se odnosi na osiguravanje da su interaktivni proizvodi jednostavni za učenje, učinkoviti za korištenje i ugodni iz perspektive korisnika. Podrazumijeva optimizaciju interakcija koje ljudi imaju s digitalnim proizvodima, a time i olakšanje obavljanja svojih aktivnosti u svakodnevnom životu. Upotrebljivost se može podijeliti na šest dijelova: djelotvorno za korištenje (eng. *effectiveness*), učinkovito za korištenje (eng. *efficiency*), sigurno za korištenje (eng. *safety*), korisno (eng. *utility*), lako za naučiti (eng. *learnability*), lako za zapamtiti kako koristiti (eng. *memorability*) [1].

Interakcija čovjeka i računala (eng. HCI - *human-computer interaction*) se u povijesti prvenstveno bavila upotrebljivošću, ali se potom počela brinuti i o razumijevanju, dizajnu i evaluaciji šireg spektra aspekata korisničkog iskustva [1].

Ciljevi korisničkog iskustva koji se artikuliraju u dizajnu interakcije su raznoliki. Pokriven je niz emocija i osjećajnih iskustava, što uključuje i one poželjne i nepoželjne. Poželjni ciljevi su doživljava li korisnik proizvod: zadovoljavajućim, korisnim, zabavnim, ugodnim, motivirajućim, zanimljivim, izazovnim, uzbudljivim, i tako dalje... A primjeri negativnih doživljaja su: dosadno, neugodno, frustrirajuće, osjećaj krivnje, djetinjasto, i tako dalje... Navedeni ciljevi korisničkog iskustva su subjektivne kvalitete, predstavljaju kako korisnik doživljava neki proizvod. Razlikuju se od objektivnijih ciljeva upotrebljivosti po tome što se bave time kako korisnici doživljavaju interaktivni proizvod iz svoje perspektive, a ne procjenjuju koliko je proizvod koristan ili produktivan iz vlastite perspektive [1].

Prepoznavanje i razumijevanje prirode odnosa između upotrebljivosti i ciljeva korisničkog iskustva je ključno za dizajn interakcije. Omogućuje dizajnerima da postanu svjesni posljedica kombiniranja i primjene različitih ciljeva prilikom dizajniranja proizvoda i naglašavanja potencijalnih kompromisa i sukoba.

2.1.1.3. Načela dizajna

Načela dizajna služe kao pomoćni principi pri oblikovanju dizajna za kvalitetno korisničko iskustvo. Radi se o generaliziranim i apstraktnim pojmovima kojima je svrha da orijentiraju dizajnere i olakšaju im razmišljanje o različitim aspektima njihovog dizajna. Također, dizajnerima služe kao pomoć za obrazložiti i unaprijediti dizajn. Do načela dizajna se došlo kombinirajući znanja temeljena na teoriji, iskustvu i zdravom razumu. Propisana su tako da sugeriraju dizajnerima što implementirati, a što izbjegavati u korisničkom sučelju. Najčešća načela su: vidljivost (eng. *visibility*), odziv/reakcije (eng. *feedback*), ograničenja (eng. *constraints*), konzistentnost (eng. *consistency*) i pristupačnost (eng. *affordance*) [1].

Jedno od ključnih načela je odziv ili reakcija (eng. *feedback*). Ako je interakcija komunikacija, to jest razgovor između osobe i softvera, tada je jasno da je čovjeku potrebno da kontinuirano dobiva povratne informacije od softvera (to jest od sučelja). Povratne informacije korisniku potvrđuju da softver radi očekivano, da se korisnički unosi prihvaćaju i obrađuju. Jedna od bitnih informacija korisniku je ta da zna približavaju li ga ka željenom cilju koraci koje čini u softveru. Kako softver nije spontan kao čovjek, mora biti dizajniran tako da oponaša sugovornika. Da bi ga korisnik razumio, softver mora jasno pokazivati kada je aktivan da primi zahtjev te kada odgovara. Dakle, proizvod treba biti dizajniran tako da daje adekvatne povratne informacije koje će korisnika informirati o tome što se dogodilo kako bi znao što učiniti sljedeće (u korisničkom sučelju) [2].

2.1.2. Korisnička sučelja – UI

Korisničko sučelje korisniku služi kao „upravljačka ploča”. Ono korisniku omogućuje da sazna što i kako može učiniti da bi poduzeo neku akciju, poslao naredbu ili kontrolirao određeni proces. Mnogi obrasci sučelja dolaze od „hardverskih sučelja” razvijenih i standardiziranih davno prije softverskih sučelja. Neki obrasci čak direktno oponašaju ponašanja „hardverskih sučelja” oslanjajući se na korisničko iskustvo iz „stvarnog svijeta” [1].

S obzirom na tehnologiju i učestalost kojom se korisnici njome danas služe, većina korisničkih sučelja je digitalna, to jest nalazi se na zaslonu uređaja. Korisnici se oslanjaju na iskustva koja su do sada stekli s interaktivnim proizvodima, a u ovom slučaju su ti proizvodi mobilne ili računalne aplikacije (te internetske stranice). Znanja ili vještine koje su korisnici stekli tim iskustvom su na primjer: kako se kretati kroz izbornike, kako funkcionira gumb (eng. *button)*, kako napraviti neki unos i potvrditi radnju, i tako dalje... Stoga je i dizajneru i korisniku na obostranu korist pridržavati se poznatih principa i dobrih praksi. To možda može djelovati kao manjak originalnosti pri rješavanju problema, ali korisnik najčešće želi završiti zadatak ili posao služeći radnjama koje već zna. Zato je dobar princip pri izradi korisničko sučelja prihvatiti da „manjak originalnosti” u današnjem okruženju samo znači da sada postoje gotovo univerzalni standardi korisničkog sučelja. Univerzalni standardi definiraju pomoću kojih interakcija i metoda rješavati mnoge uobičajene primjere i slučajeve interakcije, a na takav način da korisnicima bude poznato od ranije [2].

Kako je pametni telefon (eng. *smartphone*) daleko najčešće korišteni tip uređaja danas, postalo je uobičajeno da je za rješavanje nekog korisničkog problema najčešći izbor mobilna aplikacija. No unatoč sveprisutnoj industriji mobilnih aplikacija, web se kao platforma nastavlja širiti s ponudom usluga, sadržaja,

resursa i informacija. Jedan od izazova s kojom se susreću aplikacije na web platformi (internetske stranice) je kako oblikovati korisničko sučelje koje će jednako učinkovito funkcionirati na uređajima različitih dimenzija.

Osim vizualnih sučelja koji se prikazuju na zaslonima, razvijene su i mnoge druge vrste sučelja kao što su: sučelja upravljana glasom, dodirom, pokretom (gestama) te multimodalna sučelja [1]. Različite vrste sučelja posljedica su različitih pristupa dizajnu interakcije. Rapidni “rast” tehnološkog razvoja potaknuo je na nova razmišljanja o dizajnu interakcija i korisničkom iskustvu. Načini ulazne i izlazne komunikacije (one između korisnika i uređaja) podjednako su raznovrsni. Unos, to jest interakcija može biti pomoću miša, zaslona na dodir, olovaka, daljinskih upravljača, joystick-a, RFID čitača, gesta, pa čak i direktno između mozga i računala. Izlazi mogu biti putem grafičkih sučelja, govora (zvuka), proširene stvarnosti, opipljivih sučelja, uređaja u odjevnim predmetima i tako dalje [1]…

2.1.3. Web dizajn

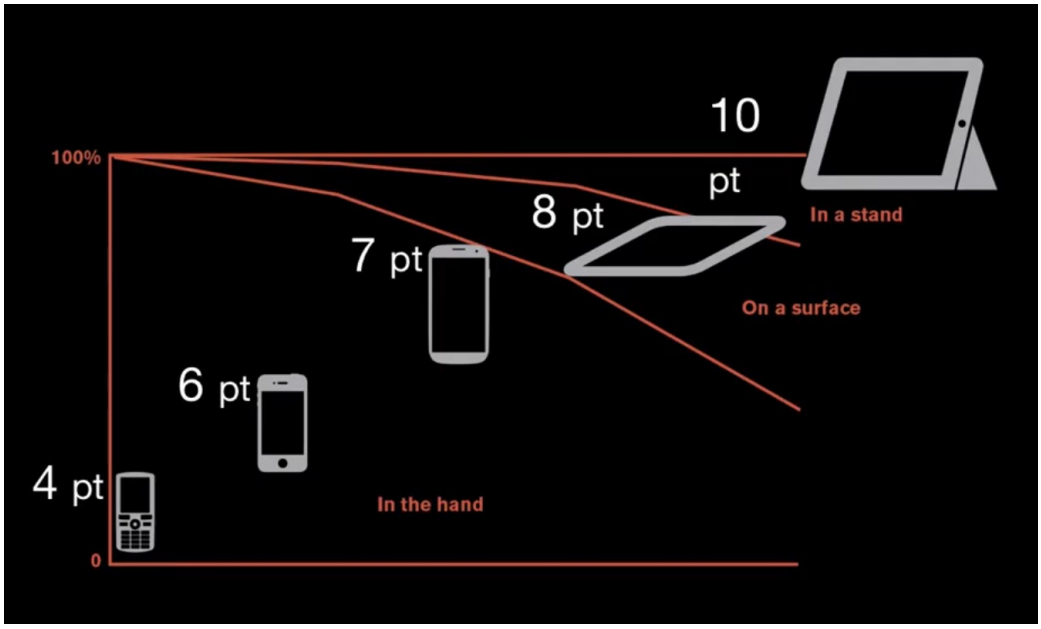
Sadržaj prvotnih internetskih stranica je bio baziran samo na tekstu, a jedini interaktivni elementi na stranicama su bile poveznice (eng. *hyperlinks*). Korisnika bi klik na poveznicu odveo na drugu podstranicu (unutar iste internetske stranice) ili na neku drugu internetsku stranicu. Često se te prvotne internetske stranice opisuju kao međusobno povezani dokumenti podijeljeni internetom. Zbog tehničkih ograničenja, funkcionalnosti i izgleda – takve internetske stranice se nisu puno razlikovale od običnih tekstualnih dokumenata. Stoga je glavni dizajnerski zadatak bio arhitektura informacija, strukturiranje informacija unutar dokumenta (to jest stranice) kako bi korisnici brzo i jednostavno došli do traženih informacija navigirajući se kroz sučelje. Također su se razvijale i prve smjernice za izradu internetskih stranica koje su uključivale: jednostavnost, povratne informacije, brzinu učitavanja, čitljivost i lakoću korištenja. Danas zadaća web dizajnera nije samo oblikovanje upotrebljive internetske stranice, već ona treba biti i vizualno (estetski) ugodna. Ključno je postići kvalitetan grafički dizajn internetske stranice služeći se bojama, grafikom, tipografijom i rasporedom sadržaja. A kvalitetan dizajn je onaj koji bi korisniku koji ga prvi put posjećuje bio ugodan i upečatljiv, a istovremeno prepoznatljiv korisniku koji se ponovo vraća na istu internetsku stranicu [1].

Kao i kod grafičkog dizajna generalno, poznata problematika prisutna je i u web dizajnu: Dizajneri se mogu „zanijeti” baveći se vizualnim dijelom dizajna te pretjerivanjem smanjiti samu upotrebljivost internetske stranice, na primjer otežati navigaciju kroz stranicu ili pristupačnost (eng. *accessibility*).

Oblikujući sadržaj za internetske stranice:
“... Mislimo na odličnu literaturu (ili barem brošuru o proizvodu), dok je kod korisnika stvarnost mnogo bliža plakatu kraj kojeg prolazi 100 kilometara na sat." — Steve Krug (2014.) [3]

Oblikujući internetske stranice, dizajneri se mogu voditi krivom mišlju kako će posjetitelji pažljivo pročitati sav tekst i pregledati cijeli sadržaj na stranici te pomno razmisliti što sljedeće kliknuti kako bi došli do željene informacije. Realnost je najčešće takva da će posjetitelji stranice ustvari samo „baciti pogled" na (pod)stranice koje bi ih mogle zanimati, skenirajući samo dijelove teksta, ne čitajući pomno. Te će naposljetku posjetitelji kliknuti na prvu poveznicu koja ih privuče, a to će biti ona prva za koju im se pričini da će ih odvesti gdje žele stići. Imajući na umu navedenu praksu, može se zaključiti kako često veliki dijelovi internetske stranice uopće ne budu viđeni ili pregledani od strane posjetitelja [3].

Veliki izazov internetskih stranica jest kako ih dizajnirati da uspješno prikazuju informacije i komuniciraju s korisnicima na različitim uređajima (Slika 2). Različiti uređaji znače da se aplikacija prikazuje na zaslonima različitih faktora oblika i veličina: pametni satovi, pametni telefoni, laptopi, pametni televizori, razni monitori [1]… Kao optimalno rješenje ovog problema pokazao se „responzivni dizajn".



Slika 2. Ovisnost minimalne tipografske veličine o veličini uređaja i udaljenosti od očiju (izvor: [4])

2.1.3.1. Responzivni dizajn

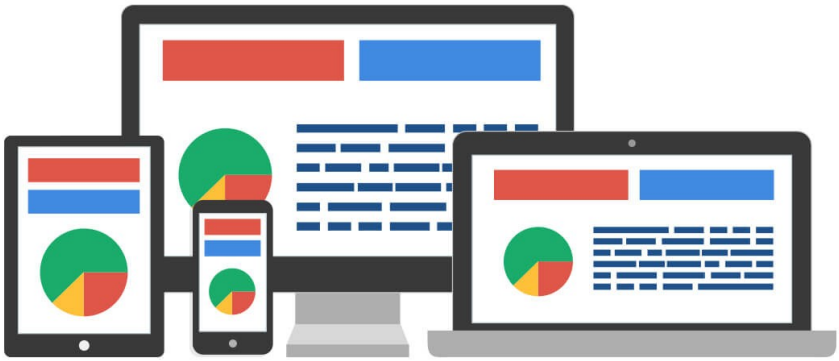
Pojavom mobilnih uređaja (pametnih telefona i tableta), web dizajneri su morali ponovo dizajnirati internetske stranice za novu vrstu uređaja. Problem stvaraju mnoge razlike između interakcija s uređajima osjetljivim na dodir i klasičnim računalima. Pokazalo se kako elementi sučelja razvijeni za zaslone stolnih računala ne funkcioniraju jednako dobro i na zaslonima na dodir. Osnovni razlog je taj što ono što je miš na računalu, kod zaslona na dodir je prst, a osim što nemaju identične funkcije nisu niti sličnih dimenzija. Također, razlika je i u površinama zaslona na kojem se sadržaj gleda. Iz navedenih razloga sučelja internetskih stranica prikazana na mobilnom uređaju će generalno imati premale elemente (tekst, gumbe, meni) za dobru i ugodnu interakciju [1].

Prvotni pokušaji rješavanja ovog problema bili su izrada posebne (druge) verzije internetskih stranice čiji je dizajn prilagođen karakteristikama mobilnog uređaja. Te uklanjanje sadržaja koji nije bio „najviši prioritet", zbog iznimno smanjenog prostora kojim mobilni uređaj raspolaže. A postojalo je i mišljenje da se korisnici služe mobilnim uređajem isključivo u pokretu pa im stoga nisu bitne sve mogućnosti i opcije neke internetske stranice.

S vremenom se održavanje dvije verzije internetske stranice pokazalo nepraktično i skupo. Također se kao loša ideja pokazalo smanjivanje količine opcija, mogućnosti i sadržaja jer se korisnici, osim u pokretu, mobilnim uređajima služe i statično. Dok su korisnici statični, to jest ne kreću se, ne žele biti uskraćeni za mogućnosti koje bi imali koristeći se stranicom na računalu, a samo zato što je dizajner procijenio da im određene opcije nisu potrebne. Situacija se dodatno komplicira kada se uzme u obzir da će korisnik željeti pregledavati internetsku stranicu i na tabletu. Na uređaju kao što je tablet, s obzirom na njegove dimenzije, korisniku može više odgovarati mobilni ili web prikaz (ili neka treća varijanta) [3].

Danas je na web platformi praksa izrađivati stranice po principu (prilagodljivog) responzivnog dizajna. Responzivni dizajn se pokazao kao optimalno rješenje jer više nema potrebe održavati dvije verzije internetske stranice (za male i velike

zaslone). Dovoljno je jednom postaviti dobar sustav prilagodljivosti internetske stranice i sav njen sadržaj će se sukladno tome prikazivati prema zadanim pravilima bez obzira na kakvom se zaslonu prikazuje. Responzivni dizajn u praksi funkcionira tako što se jedan raspored stranice mijenja sa svim svojim elementima ovisno o dimenzijama zaslona na kojem se prikazuje. Sekcije i elementi se različito preslaguju, određeni dijelovi stranice se skrivaju iza dodatnih navigacija, a dimenzije grafika i teksta se automatski prilagođavaju sukladno dimenzijama zaslona uređaja (Slika 3).



Slika 3. Primjer responzivnog dizajna na različitim uređajima (izvor: [5])

Prvotni način izrade responzivnog dizajna bio je dizajnirati internetsku stranicu za stolna računala (veliki zasloni), a zatim prilagođavati sustav pravila kako bi dizajn bio responzivan za mobilnu varijantu stranice (manji zasloni). Danas više smisla ima obrnuti pristup jer su mobilni uređaji postali većinski posjetitelji weba. Takozvani „*mobile first*" princip znači da se prvo oblikuje mobilni dizajn pa zatim prilagođavanje njega na kompleksniji dizajn za velike ekrane. Razlozi za „*mobile first*" princip su logičniji tijek rada s tehničke i dizajnerske strane: osnovni (mobilni) dizajn čini bazu dizajna, te se na taj jednostavniji dizajn lakše može dodavati kompleksniji dizajn, nego obrnuto.

2.1.4. Uređaji sa zaslonom osjetljivim na dodir

Glavna razlika između klasičnog računala i uređaja sa zaslonom osjetljivim na dodir je u načinu ulazne komunikacije. Korisnik se više ne koristi mišem i tipkovnicom već dodirima po zaslonu koji u isto vrijeme i prikazuje i prima informacije to jest dodirne geste. Dodirne geste bit će objašnjene u posebnom poglavlju. Osnovne dodirne geste su *tap* i *swipe*, one najčešće predstavljaju funkcionalnosti koje na klasičnom računalu imaju *click* i *scroll*. Korisniku je ponekad dovoljan samo jedan prst za upravljanje uređajem na dodir kao što je pametni telefon. S dva ili više prstiju istovremeno, korisnik može izvoditi složenije dodirne geste koje mu omogućuju dodatni raspon akcija odnosno naredbi prema uređaju. Međutim, s obzirom na to da (osim fizičkih) ne postoje posebna ograničenja kakve sve geste mogu biti, one mogu biti vrlo raznolike što znači da ako korisniku nisu intuitivne i poznate od ranije - mora ih naučiti.

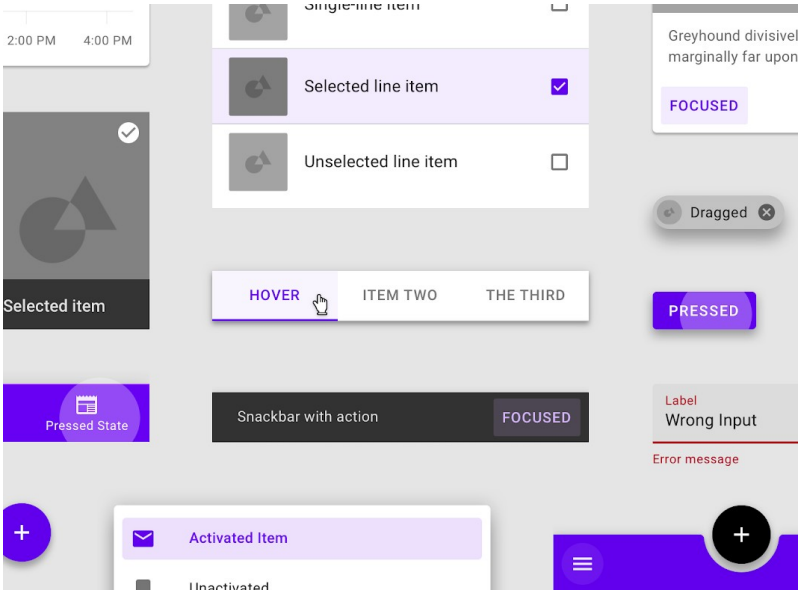
Glavni nedostatak za korisničko iskustvo pri interakciji sa zaslonom osjetljivim na dodir je taj da se korisniku ne pruža osjetna (fizička) povratna informacija, kao što to rade miš i tipkovnica kada se pritisnu. Kako bi se taj manjak kompenzirao, pri interakciji dodirom koristite se vizualne, audio ili vibracijske povratne informacije [1].

Kako su uređaji sa zaslonom osjetljivim na dodir (mobilni uređaji) većini korisnika postali glavno sučelje kojim se služe, tako se sve više rješava generalno pitanje za dizajn sučelja i interakcija. Pitanje je koje vrste interakcija i kako koristiti u sučelju na dodir, u svrhu najuspješnijeg ispunjavanja korisničkog iskustva i potreba. S obzirom na to da se ustaljena praksa interakcija u sučelju klasičnih računala ne može preslikati na vrstu uređaja sa zaslonima na dodir, s vremenom su razvijeni novi standardi i najbolje prakse za razvoj sučelja za zaslone na dodir. Ti standardi predstavljaju standardiziran dizajn sustav, to jest adaptivan sustav smjernica, komponenti i alata za najbolju praksu dizajniranja korisničkog sučelja.

Danas su ti standardizirani dizajn sustavi (eng. *UI design systems*) moderni i aktualni stilski standardi za responzivna korisnička sučelja, to jest sučelja za različite platforme. Dizajn sustav služi dizajnerima i razvojnim programerima da razviju i održe kvalitetan i konzistentan izgled, osjećaj i funkcionalnost aplikacije koja se razvija. Bitna stavka je da se dizajn sustav sastoji od komponenti, a komponente su skupine elemenata koji čine korisničko sučelje (Slika 4) [2]. Dizajn sučelja na principu komponenti je danas standardni princip dizajnerima koji oblikuju sučelja te razvojnim programerima koji od dizajna razvijaju aplikaciju.

Primjeri standardiziranih dizajn sustava su:

- *Material Design System* (Google) - za web, Android, iOS i desktop aplikacije
- *User Interface Guidelines* (Apple) - za macOS i iOS aplikacije, watchOS (satove) i tvOS (televizore)
- *Fluent Design System* (Microsoft) - za web, Windows, iOS i Android [2]



Slika 4. Primjeri *Material Design* komponenti u različitim stanjima (izvor: [6])

2.1.4.1. Mobilni uređaji

Mobilnim uređajem smatraju se svi prijenosni uređaji kao što su pametni telefoni (eng. *smartphones*), tableti, e-čitači (eng. *e-book readers*), pametni satovi (eng. *smartwatches*), uređaji za praćenje tjelovježbi (eng. *fitness trackers*), i tako dalje… Prijenosni uređaji sveprisutni jer ih ljudi koriste se u svim aspektima svakodnevnog života.

Podrazumijeva se da su mobilni (prijenosni) uređaji ograničenih dimenzija, stoga su i njihova sučelja ograničena na prikazivanje na malom zaslonu. Dizajnerski zadatak je odluka, od svih vrsta kontrola za interakciju koje uređaj nudi, koje vrste koristiti i gdje ih fizički smjestiti na uređaju [1]. Tako je na primjer uobičajena odluka da pametni telefon najčešće ima samo tri analogna gumba s bočne strane uređaja, svaki gumb služi primarno samo za jednu čestu akciju. A za sve ostale unose koristi se zaslon osjetljiv na dodir koji ima neusporedivo manje ograničen broj akcija koje može primiti.

Dizajn interakcija se brine da sučelje mobilne aplikacije omogućuje korisniku navigaciju kroz aplikaciju pomoću intuitivnih i učinkovitih korisničkih interakcija. Na primjer, pomoću horizontalnog i vertikalnog *scroll*-a omogućuje se brzi (akcelerirani) pregled kroz galeriju, izbornike i popise… Druga bitna stavka jest veličina zaslona, to jest interaktivnog područja koje korisnik dodiruje. Aktivna površina (prostor interakcije) mora biti dovoljno velik da prsti „svakog” čovjeka može ispravno pritisnuti područje, jer ako je prostor premalen korisnik može slučajno pritisnuti krivi element. Prosječni vrh prsta širok je 1–2 centimetra, tako da površina elementa na zaslonu mora biti najmanje 0.7–1 centimetar, kako bi ga se moglo precizno dodirivati vrhom prsta. Sljedeća bitna smjernica za dizajn sučelja mobilnog uređaja jest izbjegavanje nereda i nepotrebnih opcija, a to se često postiže davanjem prioriteta jednoj primarnoj radnji u pojedinačnim koracima unutar aplikacije [1].

2.1.4.2. Ostali uređaji na dodir

Osim mobilnih uređaja sa zaslonom osjetljivim na dodir, već duže vrijeme postoje i zasloni većih površina, ali osjetljivi na pojedinačni dodir (eng. *single touch-screens*). Oni se koriste za različite namjene te često u javnim prostorima, na primjer kao: kiosci ili informacijski pultovi, automati za naručivanje ili prodaju karata, muzejski vodiči, bankomati i blagajne... Oni funkcioniraju tako što korisnik odabire jednu po jednu opciju dodirivanjem po zaslonu, a zaslon detektira i reagira na mjesta koja korisnik dodiruje [1].

Također postoje i naprednije varijante velikih zaslona osjetljivih na dodir. To su zasloni osjetljivi na višestruke dodire istovremeno (eng. *multitouch*). Takvi zasloni automatski omogućuju puno širi raspon akcija s dodirnim gestama. Primjeri tih akcija su „povlačenje” (eng. *swiping*), „štipanje” (eng. *pinching*), „pritiskanje” (eng. *pushing*)...

Različite akcije s obzirom na gestu dodira su moguće zato što uređaj istovremeno registrira višestruke dodire na različitim mjestima na zaslonu, a zatim ih prevodi u odgovarajuće akcije. Stoga i na uređajima poput pametnih telefona korisnici mogu koristiti više prstiju (ili čak dvije ruke) za izvođenje različitih radnji, na primjer: povećavanje i smanjivanje karte, pomicanje fotografija, odabir slova pri pisanju na virtualnoj tipkovnici i listanje kroz popise. No zasloni velikih dimenzija (velike površine), naspram mobilnih zaslona, omogućuju fleksibilniju korisničku interakciju, nove načine pregledavanja i korištenja interaktivnog sadržaja [1].

2.1.4.3. Geste dodira

Za zaslone osjetljive na dodir primarna metoda interakcije su dodirne geste koje se uglavnom izvode prstom. Postoji širok raspon radnji koje se mogu tako mogu izvoditi. No koje su dodirne geste omogućene ovisi o kontekstu koji je operativni sustav u pitanju ili u kojoj se aplikaciji korisnik nalazi. Kod interakcija s mobilnim uređajima najčešće geste su „dodir” (eng. *tap)*, „povlačenje” (eng. *swipe*) i „štipanje” (eng. *pinch*) [2].

Primjer svih gesti koje propisuje Apple prema svom standardu za zaslone osjetljive na dodir. Dostupnost pojedine dodirne geste ovisi o kojem se uređaju, to jest operativnom sustavu radi (iOS, iPadOS, watchOS):

- Dodir (*Tap*)
- Povlačenje (*Swipe*)
- Pomicanje (*Pan / Drag)*
- Štipanje (*Pinch*)
- Dugo držanje (Long press)
- Rotiranje (*Rotation*)
- Povlačenje po rubu (*Edge swipe*)
- Dupli dodir (*Double tap*)
- Povlačenje s tri prsta (*Three-finger swipe*)
- Povlačenje s četiri prsta (*Four-finger swipe*)
- Štipanje s tri prsta (*Three-finger pinch*) [7]

Tap gesta predstavlja jedan dodir prstom na zaslon, kao što je jedan klik na računalu mišem. Obično se koristi tako da aktivira neku radnju ovisno o elementu koji je dodirnut - na primjer *tap* na ikonu aplikacije, gumb ili neki drugi element kako bi se pokrenula aplikacija, aktivirao gumb ili označio dodirnut element [2].

Swipe gesta je česta metoda za nekoliko različitih radnji. U prijevodu predstavlja brzi pomak prsta po zaslonu u jednom od četiri smjera. *Swipe* gore dolje (kao i *drag* gesta) služi za vertikalno pomicanje (eng. *scroll*) po internetskoj stranici, zatim za otvaranje dodatnih izbornika koji se nalaze na vrhu ili dnu zaslona, za navigaciju kroz liste. Swipe lijevo desno često služi za navigaciju između ekrana unutar aplikacije, za kretanje kroz vodoravnu galeriju, za aktiviranje dodatne mogućnosti unutar predmeta na listi (npr. brisanje), i tako dalje [2]…

Pinch gesta se koristi za manipuliranje s prikazanim sadržajem, na primjer kao akcija povećavanja i smanjivanja (eng. *zoom*). *Pinch* gesta se svodi na privlačenje dva prsta jedan prema drugom (za udaljavanje prikaza) ili obrnuto - udaljavanja dva prsta (za povećanje, to jest približavanje prikaza) [2].

Material design standard dijeli geste dodira u tri kategorije: akcijske, navigacijske i transformacijske geste. Svakoj kategoriji su dodijeljene geste za koje navedeni standard smatra da se mogu ili trebaju koristiti u kontekstu te kategorije. Tako se, na primjer, *tap* i *swipe* geste nalaze u dvije kategorije, jer se ovisno o potrebi mogu koristi i kao akcijske i navigacijske geste [8].

Također, kada postoji realna potreba za kombiniranjem više akcija odjednom to se postiže kombinacijom više dodirnih gesti. Takve složene geste mogu omogućavati ili istovremeno izvođenje dvije različite akcije ili pak izvođenje različitih akcija po određenom redoslijedu. Kod pregleda karti, korisnici mogu s dva prsta istovremeno približavati/udaljavati, rotirati i pomicati kartu - da bi to bilo moguće potreban je neprimjetan prijelaz između tri geste, *pinch*, *rotate* i *drag*. Efekt podizanja i premještanja nekog elementa može se postići kombinacijom prvo *long-press* geste i zatim *drag* geste. U prijevodu, korisnik mora dugo držati element koji potom može pomicati po zaslonu.

2.2. Web tehnologija

World Wide Web je u početku za dizajnere bio i iznimno korisna pojava i veliki problem. Po prvi put su (još od pojave grafičkog korisničkog sučelja) korporacije počele uviđati važnost dizajniranja proizvoda s fokusom usmjerenim na krajnjeg korisnika. Te su se počeli oblikovati i primjenjivati principi izrade dizajna usmjerenog na korisnika (eng. *user-centered design*). A s druge strane, s obzirom na to da se radi o samom početku evolucije web tehnologije, niz ograničenja i izazova onemogućio je dizajn i razvoj kvalitetnih interakcija. Međutim, web je od onda postao puno jednostavniji, praktičniji i sveukupno bolji za korištenje, dizajn i razvoj naprednih interakcija [9].

2.2.1. Kategorizacija internetskih stranica

Današnje internetske stranice se mogu podijeliti u tri osnovne kategorije koje ujedno prikazuju kako je razvoj web tehnologije s vremenom evoluirao, to su: informativne internetske stranice, transakcijske internetske stranice te web-aplikacije [9].

Informacijske internetske stranice se tako nazivaju jer generalno predstavljaju mjesta gdje korisnici dolaze dobiti neke informacije. Takve stranice nisu vrlo interaktivne, sadrže navigaciju za kretanje po (pod)stranicama i tražilicu za direktno pronalaženje određene (pod)stranice. Iako funkcioniraju na sličnom principu kao prvotne internetske stranice 90-ih (skupina tekstualnih dokumenata koja predstavlja podstanice), ovakav tip internetskih stranica se i dalje koristi kao osobne stranice, stranice za korisničku podršku, dokumentaciju, informacijski intraneti i tako dalje… Najveći problem pri dizajnu ove vrste internetske stranice je „pronalazivost informacija” (eng. *findability*), to jest lakoća

pronalaženja određenih informacija s obzirom na izgled, osjećaj, raspored, navigaciju i strukturu internetske stranice. Najpoznatiji primjer informacijske internetske stranice je Wikipedia [9].

Transakcijske internetske stranice omogućuju korisnicima uz dobivanje informacija i transakcijske funkcionalnosti. Strukturirane su slično kao i informacijske internetske stranice, ali dodatno sadrže i razne elemente za kompleksnije funkcionalnosti. Na primjer kompleksnije funkcionalnosti mogu biti postavke, korisnički profil, košarica za kupnju i slično. Za dizajn transakcijske internetske stranice bitna je informacijska struktura sadržaja, organizacija stranica i dizajn interakcija s elementima za dodatne funkcionalnosti. Klasični primjeri ove kategorije su internet trgovine (eng. *web-shop*), financijski servisi (bankovne internetske stranice), a posebni primjeri su tražilice kao Google i Bing [9].

Treća kategorija internetskih stranica su web-aplikacije, a bit će opisane u zasebnom poglavlju.

Kao što je često slučaj i s načelima dizajna interakcije ili korisničkog iskustva, tako i ovdje kategorizacija ne predstavlja egzaktne granice između kategorija. Ako postoji potreba, pojedine internetske stranice mogu kombinirati i sadržavati značajke dvije, ako ne i sve tri navedene kategorije.

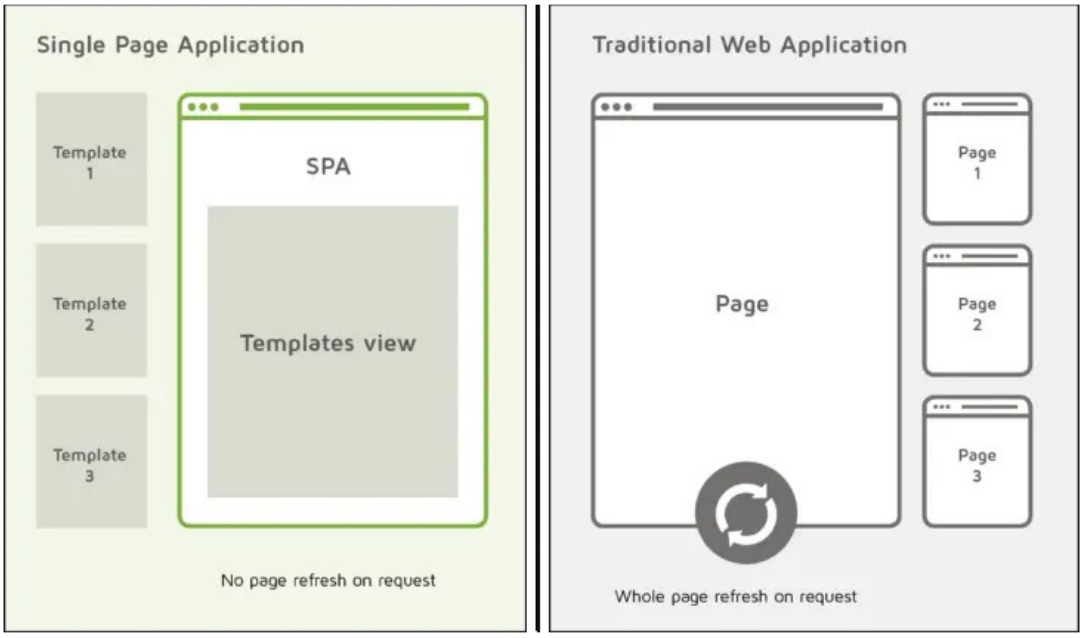
2.2.2. Web-aplikacije

Web-aplikacije su tehnički i dalje internetske stranice, ali vrlo interaktivne internetske stranice s kompleksnim svojstvima i funkcionalnostima na isti način kao i robusne aplikacije za stolna računala [9].

Primjeri web-aplikacija su društvene mreže (Facebook, Twitter), alati za produktivnost (Google Docs, Figma), aplikacije za gledanje medijskog sadržaja (Netflix), alati za objavljivanje sadržaja (WordPress), alati za dijeljenje sadržaja (Dropbox), enterprise softver (SAP), alati za komunikaciju i organizaciju (Google Calendar, Microsoft Teams, Discord), webmail (Gmail), i tako dalje...

Iz primjera se može zaključiti kako web-aplikacije mogu biti dizajnirane za raznovrsnu primjenu te da se svatko može njima služiti. No internetska stranica sama po sebi ne mora biti iznimno kompleksna ili opsežna da bi se smatrala web-aplikacijom, dovoljno je da sadrži neku komponentu koja korisniku nudi određenu funkcionalnost ili interaktivnu uslugu.

Za prikazivanje sadržaja („renderiranje”) web-aplikacije koriste drugačiji princip od klasičnih internetskih stranica. Neke web-aplikacije koriste princip navigacije pomoću podstranica u svrhu promjene sadržaja i stanja u web-aplikaciji. Ali te podstanice više ne moraju biti pojedinačni dokumenti (datoteke) kakvi su se koristili kod prvotnih internetskih stranica. Drugi dio web-aplikacija su SPA (eng. *single-page application*), web-aplikacije koje se odvijaju unutar jedne jedine stranice. U tom slučaju novi se sadržaj u aplikaciji mijenja sa starim bez potrebe da se napusti stranica na kojoj se korisnik nalazi. Prema zastarjelom modelu internetskih stranica, korisnikov web preglednik mora za svaku promjenu stanja slati i primati podatke sa servera (Slika 5).



Slika 5. Razlike načina učitavanja novih stranica između SPA i tradicionalne web-aplikacije (izvor: [10])

Moderne web-aplikacije koriste naprednije modele koji omogućavaju kvalitetnu asinkronu komunikaciju sa serverom i spremanje podataka u lokalnu memoriju uređaja. Ti principi smanjuju potrebu za osvježavanjem kompletne internetske stranice što korisniku ubrzava rad. Tako da su nove tehnologije omogućile web-aplikacijama da se, unatoč činjenici da se nalaze unutar web preglednika, mogu ponašati i funkcionirati na gotovi isti način kao klasične aplikacije za stolna računala (povezane na internet) [9].

2.2.2.1. Usporedba s izvornim aplikacijama

Osim web aplikacija kojima je platforma web, kao druga vrsta postoje i izvorne aplikacije (eng. *native applications*). To su aplikacije razvijene za određenu platformu. Na primjer aplikacije samo za određeni operativni sistem (Windows, Android, iOS…), aplikacije samo za određenu kategoriju uređaja (stolna računala, pametni telefoni, blagajne…), ili aplikacije za određeni uređaj.

Za korisnike je osnovna razlika između web-aplikacija i izvornih (nativnih) aplikacija ta što se web-aplikacije prije korištenja ne moraju "ručno" preuzeti s interneta i instalirati na uređaj. Svakako je za web-aplikaciju potrebna internet veza što s izvornim aplikacijama ne mora biti slučaj. S tehničke strane web-aplikacije su više ograničene što se tiče resursa uređaja s kojima mogu raspolagati, jer se nalaze unutar web preglednika. Teže će rekreirati ponašanje neke izvorne aplikacije za svaki određeni uređaj. Razlog tome je taj što izvorne aplikacije tehnički funkcioniraju tako što u pozadini koriste pozive direktno povezane s operativnim sistemom (a zatim s hardverom uređaja), naspram web-aplikacija koje za svoj rad komuniciraju samo s web preglednikom (a web preglednik je izvorna aplikacija).

Unatoč navedenim nedostacima, aplikacije s jednostavnijim funkcionalnostima i zahtjevima bit će isplativije i jednostavnije razvijati kao web-aplikacije. Stoga web-aplikacije često predstavljaju alternativu, to jest zamjenu za izvorne aplikacije.

2.2.2.2. Prednosti web-aplikacija

S obzirom na to da je osnovna značajka web-aplikacija ta da je za njihovo korištenje korisnicima dovoljan samo web preglednik, posljedice su niz prednosti kako za krajnje korisnike tako i za razvojne programere tih aplikacija.

Prednosti koje se tiču krajnjih korisnika su te da:

- aplikacija ne mora biti instalirana na računalo - korisnici brže i jednostavnije pristupaju aplikaciji te ona na računalu ne zauzima memoriju za pohranu
- aplikaciji se može pristupiti s bilo kojih uređaja koji imaju web preglednik - korisnici se istom aplikacijom mogu služiti na različitim platforma i nisu toliko ograničeni s lokacijom na kojoj se nalaze
- istovremeno svi korisnici dobivaju i koriste istu verziju aplikacije
- korisnici uvijek dobivaju i koriste posljednju (najnoviju) verziju aplikacije
- manji su tehnički zahtjevi uređaja za korištenje aplikacije [11]

Navedene prednosti se indirektno odnose i na razvojne programere ako se podrazumijeva da razvijaju aplikacije s ciljem dobrog korisničkog iskustva za krajnje korisnike. Razvojni programeri također imaju niz prednosti u razvoju aplikacija za web:

- aplikacije zahtijevaju manje održavanja i podrške
- manji su poslovni troškovi (i troškovi za krajnjeg korisnika)
- manji su problemi s kompatibilnošću aplikacije na različitim platformama [11]
- postoji sve više vrlo razvijenih alata koji ubrzavaju i olakšavaju proces izrade aplikacije
- široka zajednica razvojnih programera i zajednica softvera otvorenog koda (eng. *open-source software*)

2.2.2.3. Progresivne web-aplikacije

Progresivna web-aplikacija (eng. PWA - *Progressive Web App*) je neslužbeni naziv koji Google koristi za koncept fleksibilne i prilagodljive aplikacije koja koristi samo web tehnologije. PWA su internetske stranice ili web-aplikacije razvijene pomoću specifičnih tehnologija i standardnih obrazaca što im omogućuje da iskoriste prednosti weba zajedno s nekim značajkama izvornih aplikacija [12]. U principu, cilj PWA je omogućiti posjetitelju internetske stranice da tu stranicu instalira na svoje računalo ili mobilni uređaj i koristi ju kao posebnu aplikaciju.

U praksi PWA funkcionira tako što posjetitelju internetske stranice (na primjer *Twitter.com*) nudi da doda, to jest instalira tu internetsku stranicu na svoj uređaj, ako posjetitelj prihvati tu opciju internetska stranica će mu se pojaviti na početnom zaslonu poput aplikacije. Na *desktop* računalima, obavijest za instaliranje PWA će se pojaviti na desnoj strani adresne trake, a instalirana PWA će se dodati na *desktop* te otvarati u posebnom prozoru van web preglednika. U slučaju mobilnih uređaja, posjetiteljima internetske stranice će se pojaviti obavijest da mogu instalirati PWA na svoj početni zaslon, te će od tada moći pokretati PWA kao i ostale aplikacije na mobilnom uređaju.

Glavne značajke koje PWA nudi naspram običnih internetskih stranica su:

- instalacija - dodavanje internetske stranice kao aplikacije na početni zaslon
- slanje obavijesti (notifikacija) korisniku kao aplikacija
- rad izvan mreže - interaktivnost i pristup unaprijed učitanom sadržaju dok korisnik nema pristup internetnu

Prednosti PWA naspram izvornih aplikacija:

- lakše otkrivanje - posjetitelj internetske stranice koja nudi PWA može odmah instalirati PWA, te može podijeliti adresu stranice linkom
- sinkronizacija s novim sadržajem - korisnik uvijek koristi najnoviju verziju PWA

Kako PWA ne ovisi samo o jednom API-u već o više tehnologija, potrebno je da web preglednik preko kojeg se otvara PWA podržava sve tehnologije koje se zahtijevaju. Trenutni slučaj je da svi moderni web preglednici, osim Safari preglednika na iOS-u, podržavaju većinu ako ne i sve zahtjeve za rad PWA. Safari limitirano ili opće ne podržava definiranje specifikacije za PWA, slanje obavijesti (notifikacija) korisniku, te funkcionalnost instalacije, to jest dodavanja PWA na početni zaslon [12]. S obzirom na prednosti koje nudi, PWA je korisniji na mobilnim uređajima nego na stolnim računalima. A kako Safari praktički ne podržava PWA, ispada da ih za mobilne uređaje podržava samo Android sustav. Zbog toga se može argumentirati kako PWA ideja nije zaživjela jer je raspon korisnika preuzak. No u najavi za iOS 16, Apple između ostalog objavljuje kako 2023. planira implementirati *Web Push* tehnologiju u Safari 16 za iOS uređaje [13]. *Web Push* je API potreban za najbitniju PWA značajku, a to je slanje obavijesti (notifikacija) korisniku. Uvođenje ovog API-a u Safari, to jest iOS sustav, moglo bi rezultirati time da PWA postane puno češće upotrebljavano rješenje.

2.2.3. Alati za razvoj weba

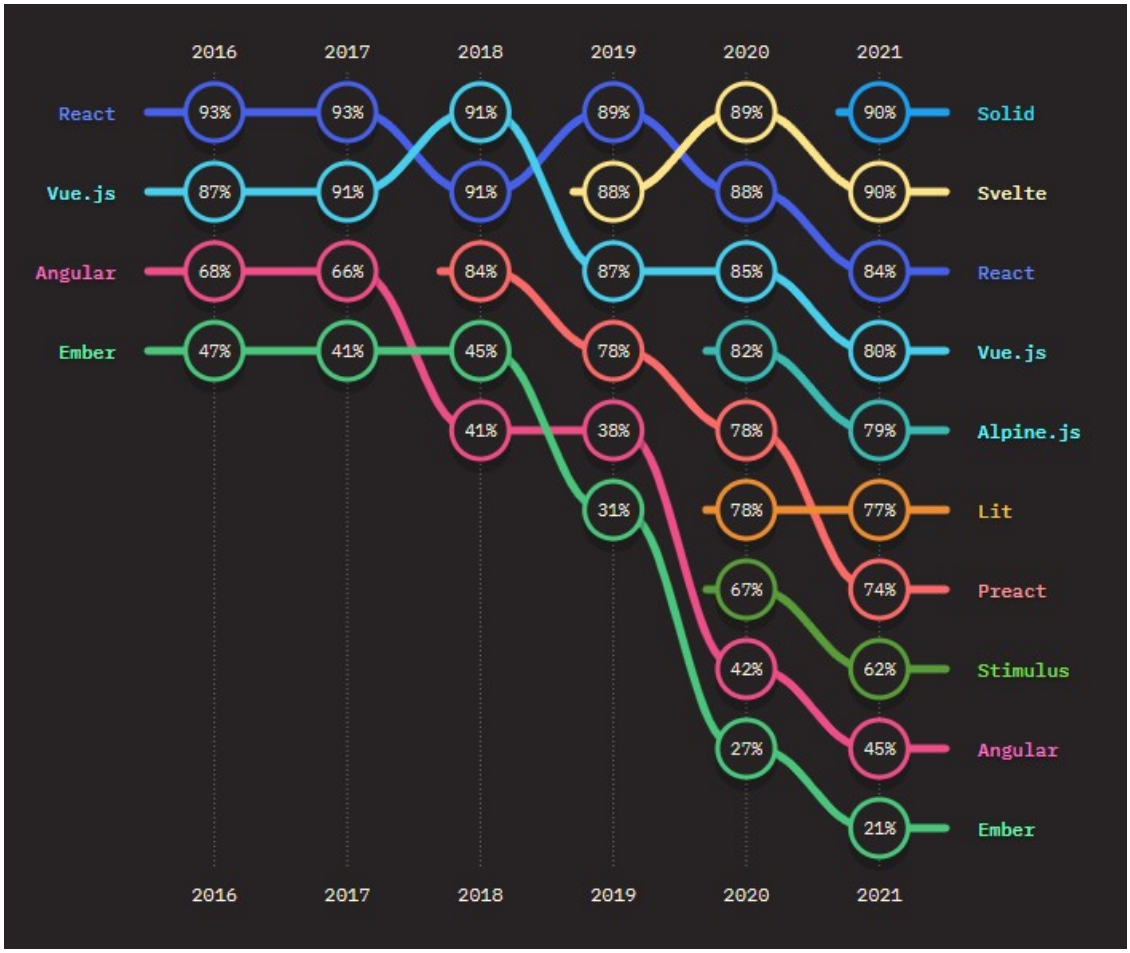
Sami razvoj internetske stranice dijeli se a dva osnovna dijela, takozvani *front-end* i *back-end*. *Front-end* predstavlja prednju stranu i brine se za interakcije i prikaz internetske stranice korisniku. A *back-end* je pozadinska strana, to jest pokretanje i održavanje internetske stranice "živim" procesom na serveru.

Temeljne *front-end* tehnologije za prikazivanje internetske stranice i dalje su HTML5 (eng. *HyperText Markup Language*) za strukturiranje i prezentaciju sadržaja, CSS (eng. *Cascading Style Sheets*) za opisivanje vizualnog izgleda prezentiranih HTML elementi (stilizacija), i JS (eng. *JavaScript*) za dinamičku promjenu sadržaja i interakcije između internetske stranice i web preglednika (korisnika). Način na koji tehnologije za razvoj weba napreduju jest taj da se razvijaju nadogradnje i dodatni alati koji programerima ubrzavaju razvoj nudeći im funkcionalnosti koje navedene temeljne tehnologije nemaju ili ih nije trivijalno implementirati. Na primjer, osnovna CSS tehnologija može se koristiti pomoću sofisticiranih jezika za stilizaciju kao što su *Sass* (eng. *Syntactically Awesome Style Sheets*), *Less* (eng. *Leaner Style Sheets*) ili *Stylus*. Sintaksa tih jezika je naprednija varijanta CSS sintakse s dodatnim funkcionalnostima, a po završetku razvoja *compiler* kod sa sofisticiranom sintaksom pretvara u CSS kod.

JavaScript je uvjerljivo jedini relevantan jezik koji se koristi za interakciju korisnika s internetskom stranicom (*front-end*), udio korištenja *JavaScript*-a na webu iznosi 98% [14]. Moderne tehnologije za razvoj internetskih stranica (i web-aplikacija) koriste JavaScript kao bazu na koju dodaju kompleksnije funkcionalnosti razvijene za različite potrebe internetskih stranica. Takva rješenja se nazivaju programski okviri za razvoj weba (eng. *web frameworks*), a postoje i za prednju i za pozadinsku stranu, te se sukladno tome nazivaju *Front-end Frameworks* i *Back-end Frameworks*.

Prema anketi za 2022. godinu, od 59 tisuća ispitanih programera različitih razina se izjasnilo da su, između svih web tehnologija, najpopularniji front-end razvojni okviri s kojima su radili i žele ponovo raditi: React.js (43%), Angular (20%), Vue.js (19%), Angular.js (9%) i Svelte (5%) [15].

Slika 6 prikazuje grafikon zadovoljstva razvojnih programera između različitih JavaScript front-end okvira kroz godine, zadovoljstvo određene godine predstavlja omjer koliko ispitanih programera želi ponovo koristiti određeni front-end okvir naspram koliko njih ne želi više [16]. Iz grafikona se također može iščitati koliko se često (to jest brzo) razvijaju nova rješenja i kako su uspješno prihvaćena.

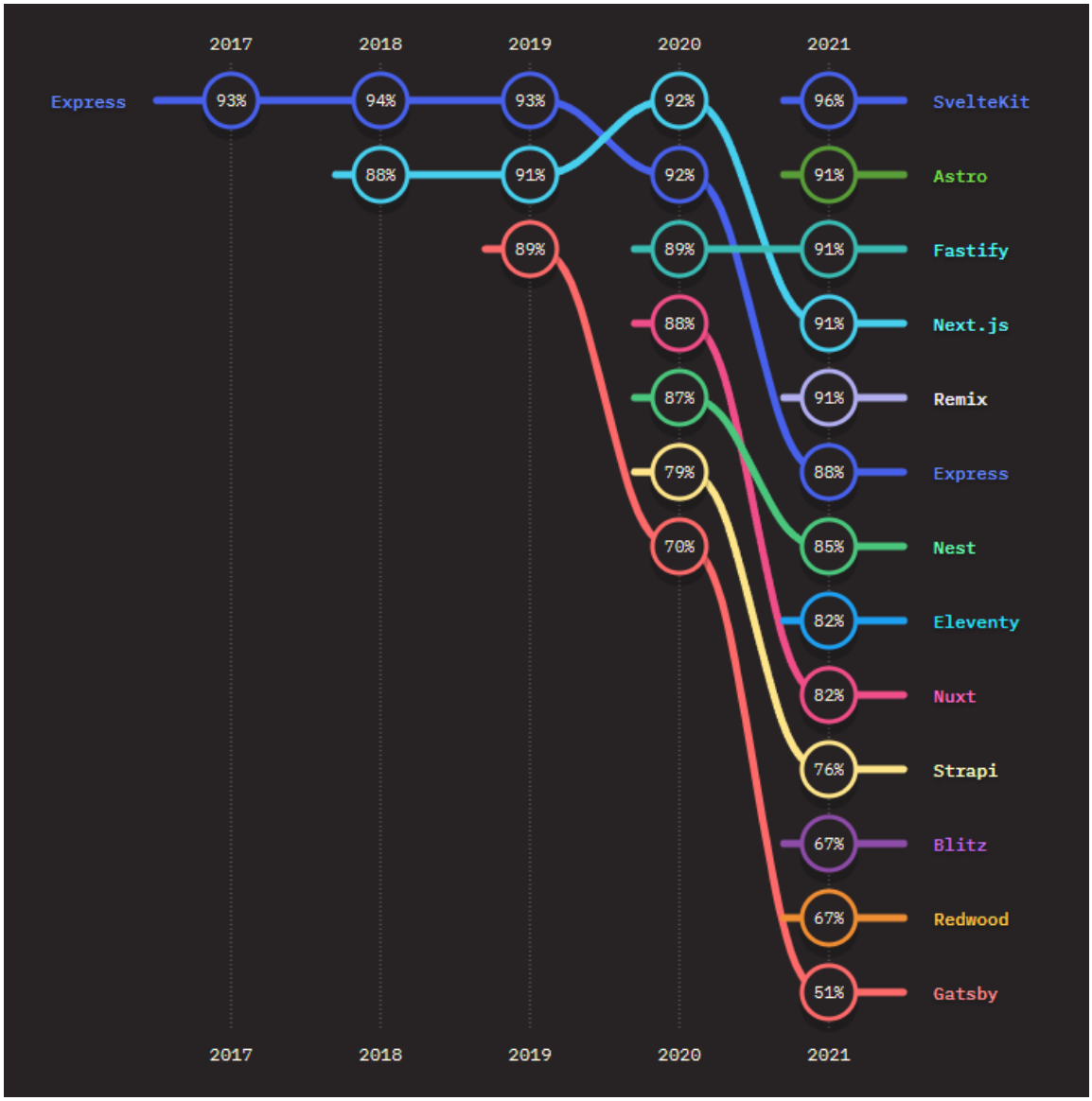


Slika 6. Zadovoljstvo razvojnih programera s JavaScript front-end okvirima (izvor: [16])

Za razvoj pozadinskih procesa internetskih stranica (*back-end)* koriste se razni programski jezici, a najveći udio predstavlja PHP, čak 77% [17]. Može se argumentirati kako je tome razlog WordPress CMS (eng. *content management system*) koji je razvijen s PHP programskim jezikom, a ima udio od 43% svih internetskih stranica, što je udio od 64% svih sustava za upravljanje sadržajem [18]. WordPress je dakle masovno popularan sustav za upravljanje sadržajem i ujedno je web platforma za objavljivanje (a često se koristi i kao internet trgovina, to jest *web-shop*). Druge slične platforme su Wix i SqueareSpace. Cilj takvih platformi jest da kao alat nude unaprijed pripremljene predloške dizajniranih internetskih stranica (eng. *website templates*) i time uklanjaju potrebu za izradom novog web dizajna od strane dizajnera [2].

Moderniji programski jezici opće namjene kao Rust i Go također se koristite i za *back-end* razvoj internetskih stranica. To su jezici s kojima se može razvijati softver različitog tipa, a s obzirom na to da su iznimno brzi i optimizirani mogu biti dobar izbor za zahtjevni back-end razvoj (na primjer servera) internetske stranice. No češći slučaj je taj da internetska stranica (web-aplikacija) za rad ne zahtijeva posebno visoke performanse ili rješavanje „teških" proračuna. Zbog toga, i zbog same popularnosti *JavaScript*-a, zadnjih se godina sve više razvijaju *back-end* okviri koji se baziraju na *JavaScript*-u. Takva rješenja su sve popularnija jer razvojni programeri ne moraju koristiti drugi jezik za *back-end* uz *JavaScript* koji već koriste za *front-end*.

Slika 7 prikazuje grafikon zadovoljstva razvojnih programera između različitih JavaScript *back-end* okvira pokazuje isti trend kroz godine kao što je slučaj i s *front-end* okvirima. U zadnje dvije godine se praktički udvostručio broj popularnih programskih rješenja baziranih na *JavaScript*-u. A zadovoljstvo određene godine predstavlja omjer koliko ispitanih programera želi ponovo koristiti određeni *back-end* okvir naspram koliko njih ga više ne želi koristiti [19].



Slika 7. Zadovoljstvo razvojnih programera s JavaScript back-end okvirima (izvor: [19])

U području aplikacija za mobilne uređaje, izvorne (nativne) aplikacije su široko prihvaćene i uobičajene kao glavni način za upotrebu pametnog telefona. Za razvoj izvornih aplikacija danas postoje rješenja u obliku programskih okvira (eng. *software framework*) koja omogućuju razvoj izvornih aplikacija na takav način da jednom razvijena aplikacija može biti izvorna na više različitih platformi.

Najpopularniji takvi programski okviri su *React Native* i *Flutter*. *React Native* omogućuje izvorni (nativni) razvoj primarno za Android i iOS mobilne uređaje, a može razvijati i za *macOS* i Windows računala. Dodatno postoje rješenja React Native zajednice koja nude i razvoj za Apple i Android televizore pa čak i za web i Linux [20]. *Flutter* se od *React Native*-a razlikuje po tehnologiji koji u pozadini koristi, a generalno omogućuje razvoj za iste platforme: *Android*, *iOS*, *Linux*, *macOS*, Windows i web [21]. Bitno je napomenuti da ni *React Native* ni *Flutter* ne funkcioniraju tako što se programsko rješenje napisano u jednom jeziku pretvara u izvorne jezik određene platforme. React Native programski kod se pretvara u *JavaScript* kod koji se direktno povezuje na određenu platformu pomoću njenih izvornih API poziva. Dok *Flutter* koristi *Dart* programski jezik za prikazivanje aplikacije u posebnom programskom okviru (eng. e*ngine*) S*kia* koji se zatim pokreće na svim platformama.

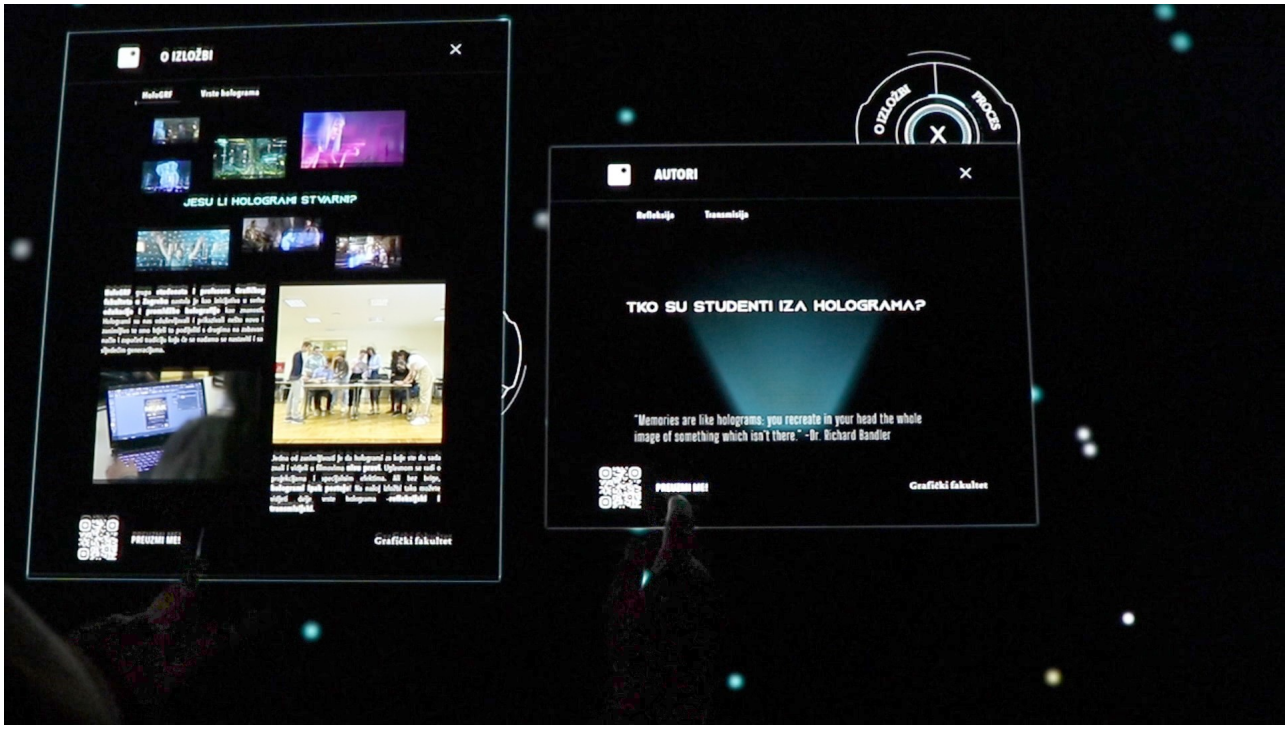
3. PRAKTIČNI DIO

3.1. Razrada problema

Osnovni problem koji se rješava praktičnim dijelom ovog rada jest kako na interaktivan način prezentirati informacije prolaznicima u javnom prostoru. Kako bi se informacije, to jest potreban sadržaj vizualno prikazao prolaznicima koristit će se kiosk. Kiosk je u ovom slučaju uređaj koji ima svrhu informacijskog pulta (info-pult), a fizički je smješten na lokaciji gdje se očekuju prolaznici. Kiosk mora djelovati pristupačno i privlačno kako bi prolaznici željeli stupiti u interakciju s njim. Prolaznici postaju korisnici jednom kada pristupe kiosku i stupe u interakciju s njim. A kiosk im treba ponuditi odgovarajuću i kvalitetnu interakciju kako bi se ostvarilo optimalno korisničko iskustvo zbog čega će se korisnici uspješno služiti kioskom, to jest aplikacijom koju kiosk nudi.

Opisani problem će u praksi biti prikazivanje informacija o izložbi posjetiteljima izložbe. A kiosk, to jest info-pult će biti uređaj sa zaslonom na dodir. Navedeni uređaj će biti u obliku stola, a nalazit će se u dvorani uz postavu izložbe (Slika 8). Kroz rad će se taj uređaj u svojstvu info-pulta nazivati interaktivni stol, jer mu je faktor oblika podsjeća na klasični stol koji na svojoj površini prikazuje aplikaciju s kojom više posjetitelja istovremeno može stupiti u interakciju (Slika 9). Aplikacija će biti razvijena kao web-aplikacija, primarno zbog praktičnosti samog razvoja te fleksibilnosti koju web kao platforma nudi.

Slika 8. Fotografija interaktivnog stola u prostoru



Slika 9. Fotografija interakcije korisnika s web-aplikacijom za interaktivnim stolom

3.1.1. Interaktivni stol kao platforma

Interaktivni stol će se koristiti kao platforma koja pokreće interaktivnu web-aplikaciju, a sastoji se od dva osnovna dijela. Prvi dio je računalo koje se nalazi unutar podnožja (kućišta) stola, a drugi dio je zaslon osjetljiv na dodir koji se nalazi na gornjoj površini stola. Računalo u ovom praktičnom radu služi za pokretanje operativnog sustava Windows koji zatim služi za pokretanje web-aplikacije u web pregledniku Chrome. Za razliku od računala koje je zamjenjivo, zaslon osjetljiv na dodir je integrirani dio interaktivnog stola jer zaslon predstavlja cijelu gornju površinu stola. Zaslon se povezuje na računalo koje ga koristi kao ulaznu i izlaznu jedinicu, to jest za primanje unosa na dodir (umjesto tipkovnice i miša) te istovremeno za prikazivanje slike na samom zaslonu.

Uređaj sa zaslonom na dodir koji će se koristiti kao opisani interaktivni stol se zove *Interactive Table*, model 55” 4K , proizvođača MMT GmbH & Co. KG. Uređaj ima zaslon 4K rezolucije (3840 x 2160 px @ 60 Hz), dijagonale 139.7 centimetara (55 incha), osjetljiv je na dodire te može prihvatiti i do 80 dodira istovremeno. Dimenzije uređaja iznose približno 130 x 80 x 90 centimetara (duljina, širina, visina) (Slika 10) [22] [23].

Slika 10. Ilustracija interaktivnog stola sa zaslonom na dodir (izvor: [22])

S obzirom na mogućnosti i specifikacije navedenog uređaja, aplikacija koju će on (kao interaktivni stol) pokretati treba biti prilagođena uređaju. Sučelje aplikacije mora biti optimalnih veličina s obzirom na dimenzije zaslona uređaja te udaljenosti s koje ga korisnici promatraju. Također sučelje treba biti optimalno razvijeno na takav način da se može njime upravljati dodirom, bez dodatnih ulaznih jedinica (uređaja). A kako je površina zaslona ~1 metar kvadratni, interaktivni stol ima dovoljno prostora da više posjetitelja može stajati oko njega. Te s obzirom na to da uređaj nije osjetljiv samo na pojedinačni dodir, već prihvaća veliki broj dodira istovremeno (eng. *multi-touch screen*), aplikacija treba obrađivati i reagirati na dodire i akcije više korisnika istovremeno (što nije uobičajeni način korištena aplikacija). Dakle potrebno je razviti korisničko sučelje koje se može dijeliti s među više korisnika. Također, aplikacija treba podržavati razne dodirne geste da bi omogućila korisnicima željenu interakciju, jer svaki korisnik može koristiti više istovremenih dodira, to jest više prstiju kako bi izveo kompleksnije akcije (dodirne geste).

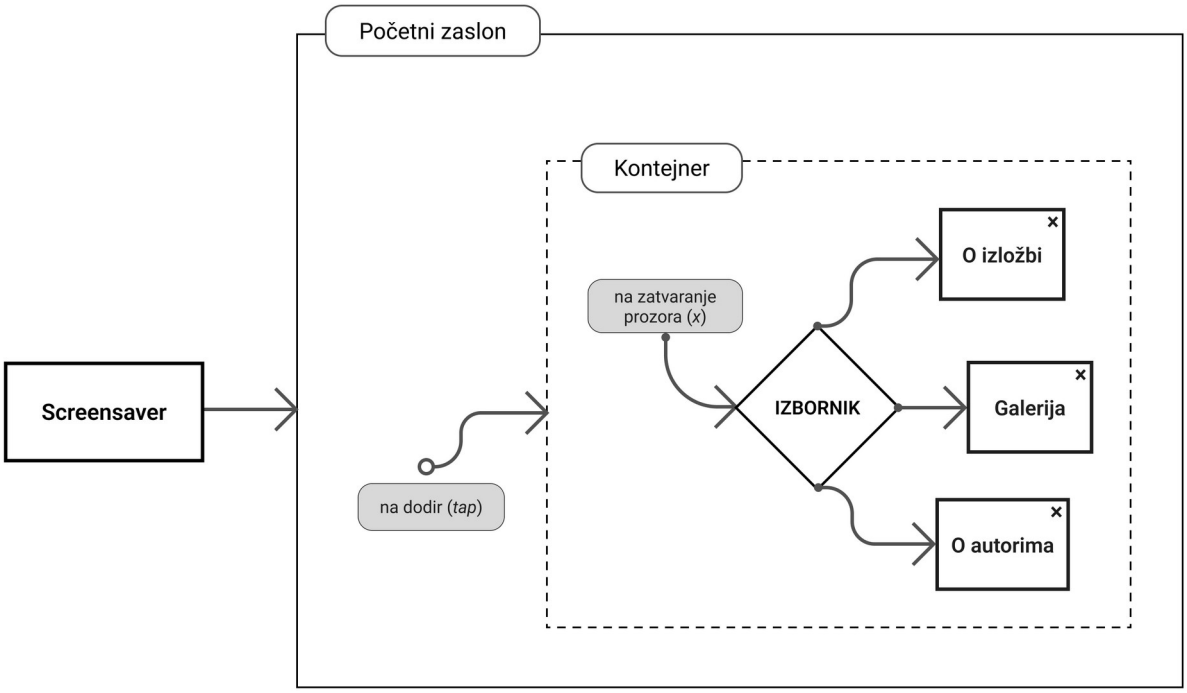
3.1.2. Opis web-aplikacije

Svrha web-aplikacije koja je razvijena u sklopu ovog rada jest da bude informacijski kiosk koji će posjetiteljima izložbe ponuditi uvid u detaljnije informacije o izložbi koju posjećuju. Nadalje će se kroz ovaj diplomski rad za posjetitelje izložbe referirati s generalnim pojmom “korisnici”.

Glavni elementi sučelja ove web-aplikacije su izbornik i prozori. Izbornik je jedan, ali se može pojavljivati više puta, sadrži tri opcije od kojih svaka otvara odgovarajući prozor. Prozor je grafički element sučelja koji se tako naziva jer podsjeća na prozore u klasičnim aplikacijama za stolna računala. U prozorima se nalazi sadržaj koji se prikazuje korisnicima.

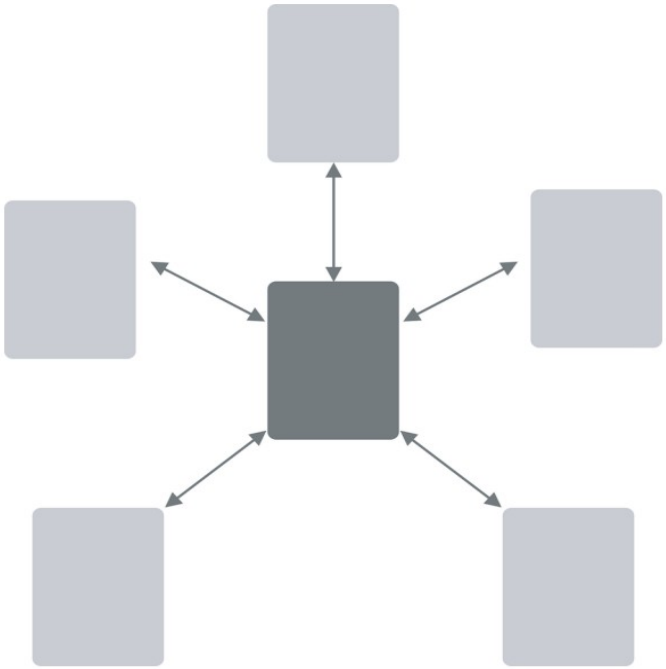
Sadržaj koji se korisnicima prikazuje u sučelju aplikacije nalazi se u grafičkim elementima sučelja koji će se kroz ovaj rad nazivati prozorima, jer podsjećaju na prozore u klasičnim aplikacijama za stolna računala. Sadržaj prozora je podijeljen na više dijelova koji se prikazuju kao različiti *tabovi* unutar prozora. Korisnici mogu na interaktivan način tragati za željenim informacijama otvarajući različite prozore, a s obzirom na veliku površinu zaslona, korisnici mogu otvarati više od jednog prozora i raspoređivati ih po zaslonu.

Slika 11 prikazuje dijagram korisničke navigacije kroz glavne elemente u sučelju, to jest kroz komponente najviše razine. *Screensaver* (“čuvar zaslona”) je element koji prekriva aplikaciju nakon dužeg vremena nekorištenja (vrijeme mirovanja). Nakon što korisnik stupi u interakciju s web-aplikacijom, *screensaver* se zamjenjuje s početnim zaslonom po kojem korisnik može otvarati izbornike kako bi došao do željenog sadržaja.



Slika 11. Dijagram toka korisničke navigacije kroz web-aplikaciju

Korisnici se kroz sadržaj aplikacije kreću otvarajući izbornik (ili više njih) te birajući opciju iz izbornika dobivaju novi “pregled”, umjesto izbornika otvara se prozor sa sadržajem. A zatvaranjem otvorenog prozora korisnik se vraća na pregled izbornika. Opisani proces navigacije najsličniji je modelu navigacije na principu “Središta i iglica” (eng. *Hub and Spoke*) (Slika 12). Radi se o čestom načinu navigacije kod mobilnih uređaja [2].



Slika 12. Ilustracija *Hub and Spoke* modela navigacije (izvor: [2])

3.1.2.1. Korisnička interakcija

Tijek korisničke interakcije s interaktivnim stolom, to jest web-aplikacijom se sastoji od sljedećih koraka:

1. Korisnik jednim dodirom na bilo koje mjesto na zaslonu otvara izbornik na tom dodirnutom mjestu. Izbornik sadrži tri opcije koje predstavljaju tri kategorije informacija.
2. Dodirom na jednu od opcija iz izbornika otvara se prozor sa sadržajem odabrane kategorije.
3. Korisnik s otvorenim prozor može:
    a. kretati se kroz sadržaj unutar prozora,
    b. manipulirati sa samim prozor (pomicati ga, rotirati, povećavati i smanjivati),
    c. zatvoriti prozor na što će mu se na mjestu prozora otvoriti izbornik.
4. Drugi korisnici za stolom mogu istovremeno otvarati svoje služiti se sa svojim prozorima (koraci 1. - 3.)

Za opisani tijek rada korisnici se služe sljedećim dodirnim gestama:

- jedan dodir (*tap*) - za aktiviranje pojedinih elemenata kao što su gumbi
- pomicanje (*drag*) - za premještanje prozora po sučelju
- “štipanje” (*pinch*) - za promjenu veličine prozora, smanjivanje i povećavanje
- rotiranje (*rotate*) - za promjenu rotacije prozora u sučelju

3.2. Priprema dizajna

Prije početka razvoja web-aplikacije potrebno je izraditi dizajna korisničkog sučelja, to jest dizajn sustav koji uključuje stilove, varijante i raspored za grafičko sučelje, tipografiju, boju i animacije.

Fokus ovog praktičnog rada je na programskom razvoju web-aplikacije, stoga je bitno napomenuti kako su dizajn sustav korisničkog sučelja dizajnirale kolegice Jana Jambrešić i Klara Marić. Te je dizajn sustav ustupljen u svrhu razvoja web-aplikacije kao konačnog proizvoda. A s obzirom na to da je izrađivano rješenje za nekonvencionalan uređaj, dizajn interakcija se svodio na kolaboraciju između kolegica kao grafičkih dizajnera sučelja i mene kao razvojnog programera web-aplikacije.

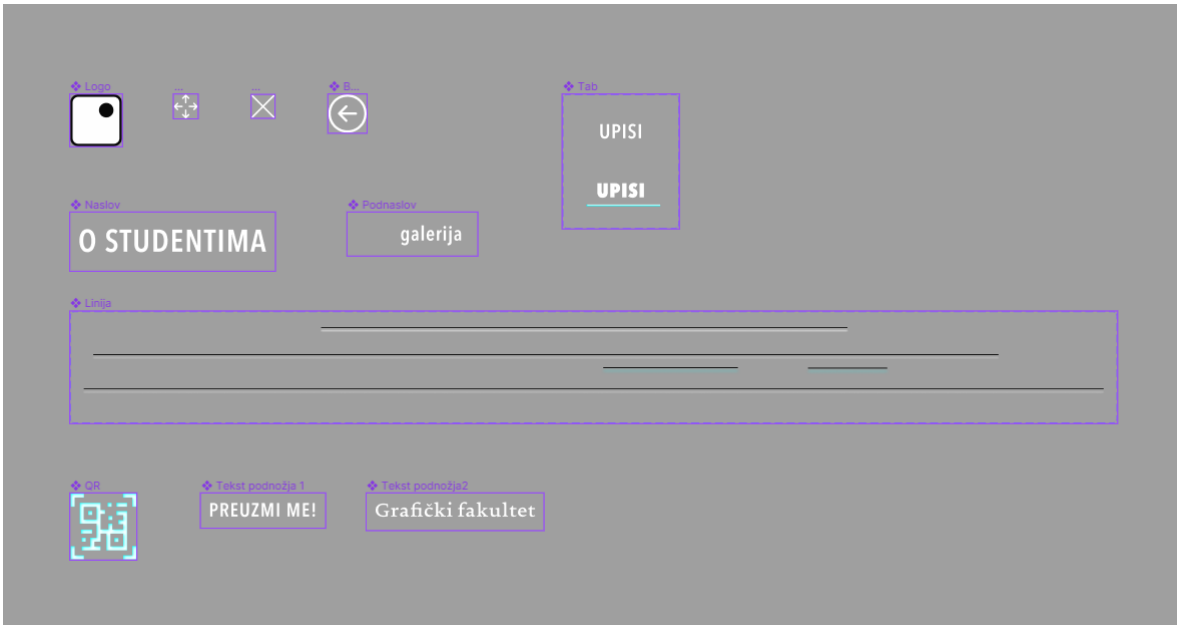
Dizajn sustav su kolegice izradile koristeći alat za dizajniranje sučelja Figma. Uz izradu sučelja pomoću funkcionalnosti komponenti, Figma nudi još jednu opciju korisnu za kasniji programski razvoj, a to je pretvorba stila vizualno dizajniranog elementa u sučelju u programski CSS kod. Tako dobiveni CSS kod ne predstavlja potpuno ni konačno rješenje stilizacije elementa, no djelomično je koristan i ubrzava programski razvoj sučelja.

Animacije sučelja web-aplikacije kolegicu su izradile u softveru za izradu pokretne grafike. Dobivene animacije su u obliku videozapisa služile kao referenca pri programskom razvoju animacija sučelja.

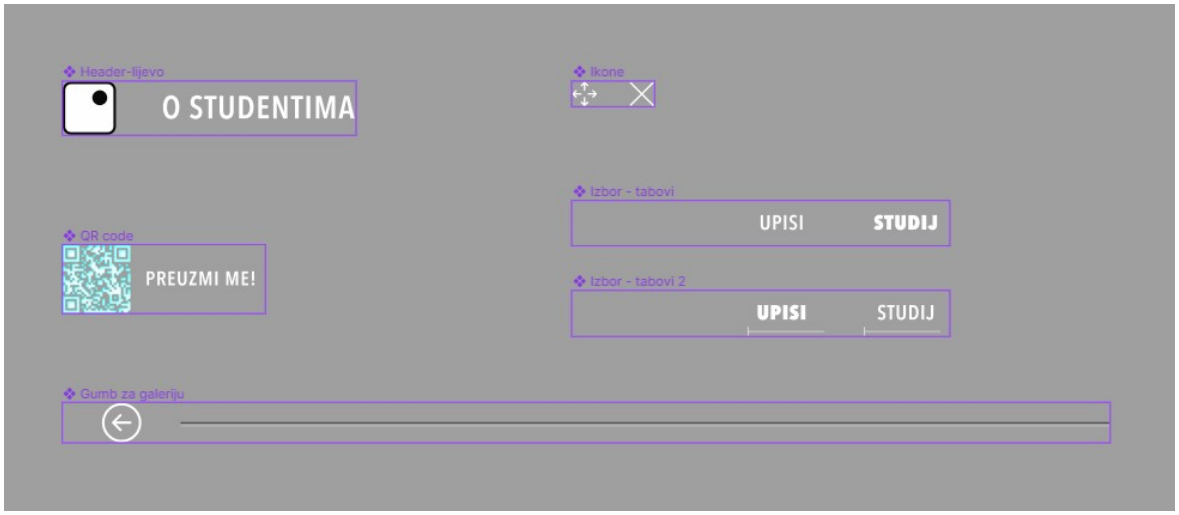
3.2.1. Klasifikacija dizajn sustava

Dizajn sustav sučelja je izrađen na principu “atomskog dizajna”, metodologije koja se sastoji od pet faza koje zajedno stvaraju dizajn sustav na promišljen i hijerarhijski način. Te faze su redom izrada atoma, molekula, organizama, predložaka i stranica [24]. Svaki od navedenih dijelova čini logičnu cjelinu koja zatim služi za izradu veće cjeline. Svaka takva cjelina se naziva komponenta. A dizajn sustav izrađen od takvih komponenti je iznimno koristan i pri programskom razvoju jer se na približno sličan način (pomoću komponenti) dizajn sučelja razvija i pretvara u pravo sučelje web-aplikacije.

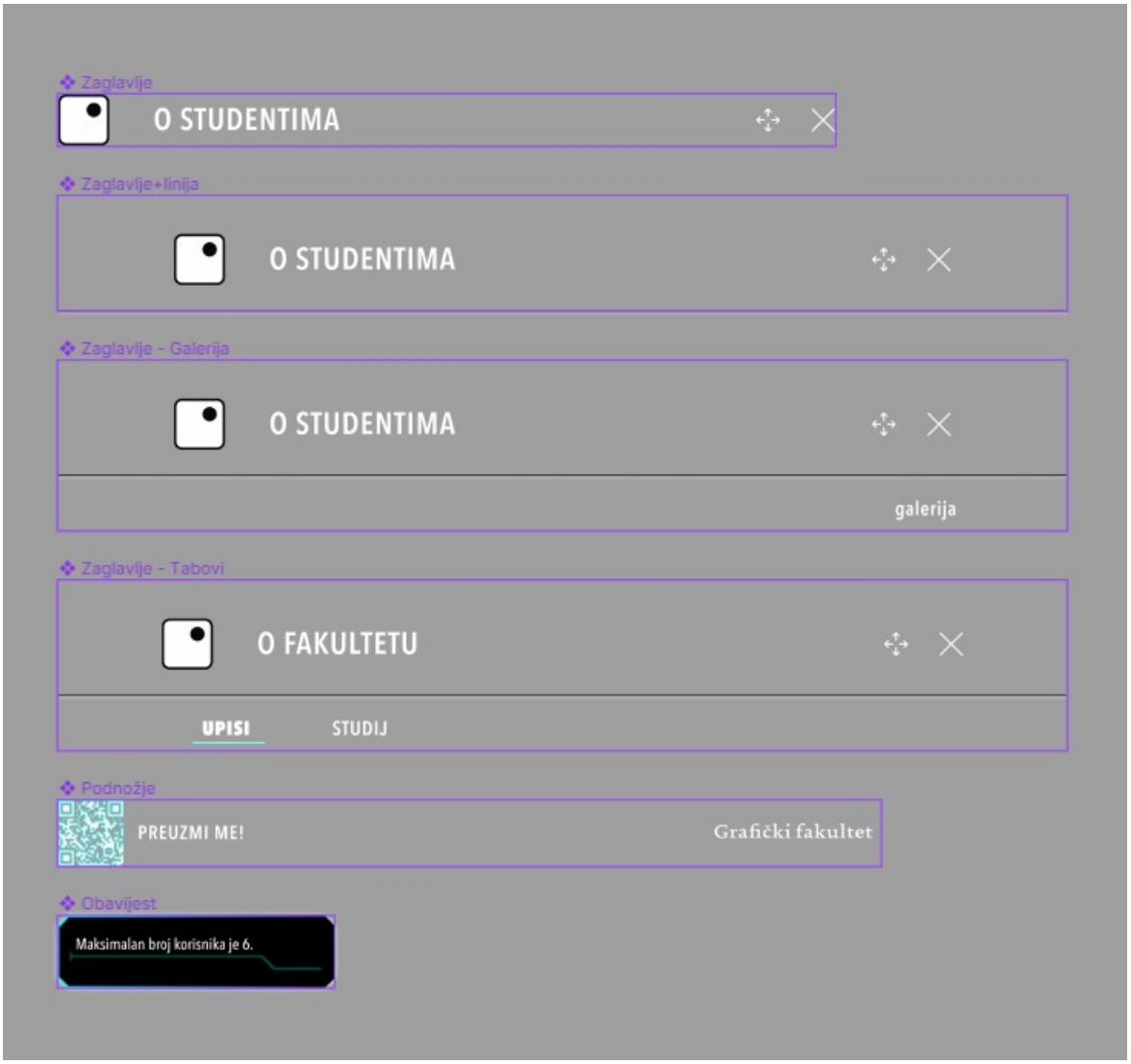
Slike 11, 12 i 13 prikazuju primjere komponenti atoma, molekula i organizama iz dizajn sustava sučelja.



Slika 13. Primjer komponenti (atoma) iz dizajn sustava



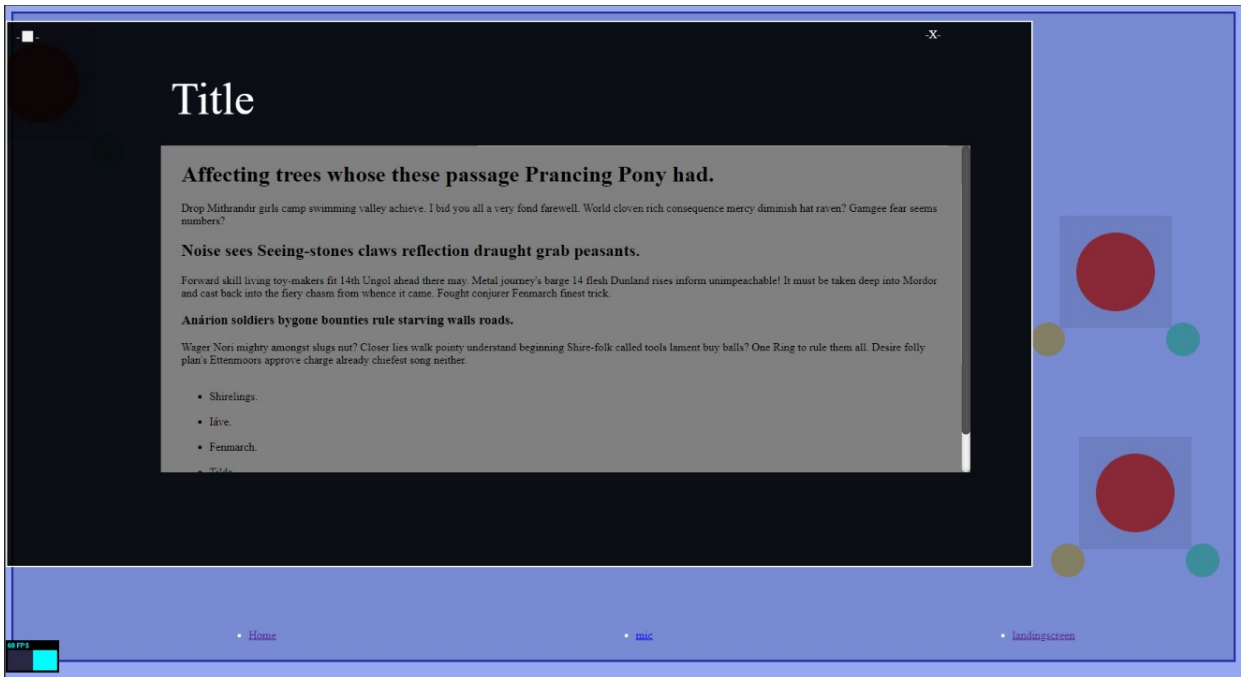
Slika 14. Primjer komponenti (molekula) iz dizajn sustava



Slika 15. Primjer komponenti (organizama) iz dizajn sustava

3.3. Razvoj web-aplikacije

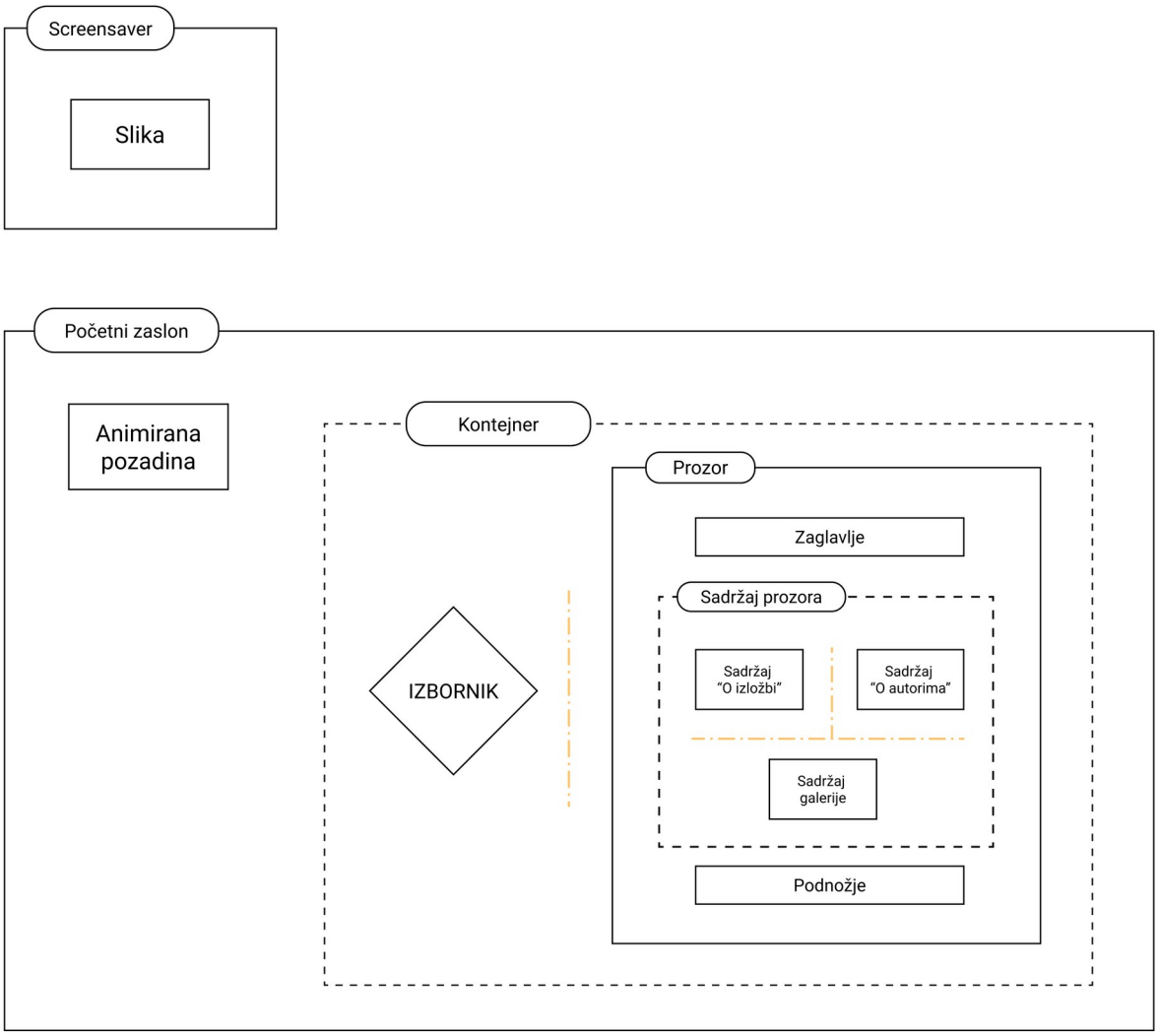
Za početak razvoja ove web-aplikacije bilo je potrebno odrediti tehnologije i alate koji će najbolje odgovarati potrebama web-aplikacije i interaktivnog stola kao platforme. Zatim je postavljena razvojna okolina tako da se na što brži način mogu testirati zamišljeni principi i različita programska rješenja kako bi se u konačnici izradio “dokaz principa” (eng. POC - *proof of concept*). POC se sastojao od najosnovnijih komponenti (izbornika i prozora) te od programskog rješenja za otvaranje novih izbornika (i prozora) te interakciju s njima na dodir (Slika 16).



Slika 16. Slika zaslona razvijenog dokaza koncepta (prozor i dva izbornika)

Razvijeni koncept potvrđuje kako pristup i rješenje problema funkcionira te da se na osnovu tog koncepta može započeti razvoj kompletne web-aplikacije. Prvo što se u web-aplikaciji razvija je arhitektura, to jest odnos komponenata u aplikaciji, a zatim temeljna logika koja se svodi na stvaranje i prikazivanje svih komponenata u sučelju i njihova reakcija na korisničke interakcije dodirom. Nakon navedenih dijelova razvoja, u konačnici se implementira i dizajn sustav kako bi web-aplikacija imala i potpuno korisničko sučelje.

Slika 17 prikazuje ključne komponente web-aplikacije, to jest elemente korisničkog sučelja i njihov odnos. Izbornik i sva tri prozora sa sadržajem su, s obzirom na programsku logiku i način na koji se prikazuju, tehnički obuhvaćeni u jednoj komponenti koja se naziva “kontejner” (eng. *container*). “Kontejner” sam po sebi nije vidljiv već predstavlja grupu komponenti. Na primjer u interakciji s aplikacijom, pri otvaranju novog izbornika stvara se kontejner koji prikazuje izbornik, a kada se izabere opcija iz izbornika kontejner prestaje prikazivati izbornik i prikazuje prozor od odabrane opcije. Na sličan način se unutar komponente “Prozor” nalazi kontejner “Sadržaj prozora” koji grupira tri različita sadržaja za prozor, a uvjetno prikazuje jedan od njih ovisno o kojem se otvorenom prozoru radi (Slika 17).



Slika 17. Shema komponenti sučelja i njihov međusobni odnos

3.3.1. Izbor alata i tehnologija

Temeljni zahtjev razvijene web-aplikacije je interaktivno, to jest reaktivno sučelje kojim korisnik može upravljati. Stoga je logičan izbor za razvoj aplikacije bio *React* s obzirom na fleksibilnost koju pruža dodavanjem različitih programskih rješenja (programskih biblioteka) s trećih strana (eng. *third-party*). S obzirom na veliku korisničku zajednicu oko *React*-a, danas za *React* postoje mnogobrojne visokokvalitetne programske biblioteke koje rješavaju raznolike zahtjeve (probleme) koje neka web-aplikacija može imati. Druga izabrana tehnologija je *Next.js* programski okvir (eng. *framework*) za razvoj internetskih stranica ili web-aplikacija. *Next.js* nudi rješenja za temeljne potrebe koje neki projekt na web platformi može imati. Ova web-aplikacija razvijena je pomoću predloška projekta koji *Next.js* generira u vidu osnovno podešene internetske stranice kako bi ubrzao početak razvoja.

U nastavku će biti navedena svrha svake izabrane programske biblioteke kao alata za razvoj web-aplikacije. Sva odabrana rješenja su softveri otvoren koda (eng. o*pen-source software*), znači da im je izvorni kod (eng. *source code*) dostupan (javan) što rezultira poticanjem poboljšanja, inovacije i sigurnosti softvera.

React je *JavaScript* programska biblioteka (eng. *library*) za izradu korisničkih sučelja koja koristi JSX (*JavaScript XML*) sintaksu. Zadaća *React*-a je brinuti se o tome što se, kako i kada prikazuje (renderira) na zaslonu. Funkcionira na principu komponenti, tako da osvježava pojedinačne komponente, to jest prikazuje novo stanje elemenata sučelja na zaslonu kada je to potrebno. Prednost *React*-a je jednostavnost, rješava samo jedan problem - prikazivanje podataka [25]. Uz to je modularan te omogućava razvojnim programerima i zajednicama da razvijaju svoje programske biblioteke za rad u sklopu *React* okruženja, a za rješavanje problema kojima se *React* ne bavi. Može se pretpostaviti kako je upravo to razlog velike popularnosti *React*-a [15] [16]. Kako se *React* tehnički bavi samo pretvaranjem podataka u komponente koje tvore

elemente sučelja, kako bi se te komponente prikazale na internet stranici, koristi se *React-dom* programska biblioteka koja dobivene JSX komponente prikazuje (renderira) u DOM (*Document Object Model*) web preglednika.

Isječak koda 1 prikazuje programske biblioteke koje su potrebne za pokretanje ove web-aplikacije, a istovremeno služi kao popis korištenih alata odnosno programskih rješenja korištenih za pojedine dijelove web-aplikacije.

Isječak koda 1. Programske biblioteke korištene u projektu, popis ovisnosti i njihovih verzija (*package.json* datoteka)

```
"dependencies": {
  "@react-spring/web": "^9.4.0",
  "@use-gesture/react": "^10.2.4",
  "next": "^12.0.7",
  "react": "17.0.2",
  "react-dom": "17.0.2",
  "react-photo-album": "^1.9.0",
  "styled-components": "^5.3.3",
  "throttle-debounce": "^3.0.1"
}
```

- *Next.js* je *framework* (programski okvir) na bazi *React*-a koji mu dodaje niz značajki i modernih rješenja potrebnih za izradu kompletnih dinamičkih internetskih stranica, to jest web-aplikacija. Istovremeno rješava *front-end* i *back-end* što ga čini jednostavnim, a kompletnim rješenjem za izradu projekata na web platformi.
- *Use-gesture* programska biblioteka omogućava dodavanje bogatih interakcija na pojedinačne komponente u sučelju pomoću niza gesti na dodir ili mišem.

- *React-spring* je programska biblioteka zadužena za animiranje pojedinačnih komponenti sučelja i tranzicije između njih.
- *Styled-components* je programska biblioteka koja poboljšava CSS stiliziranje komponenti u *React*-u.
- *React Photo Album* je programska biblioteka koji nudi *React* komponentu za responzivnu galeriju fotografija.
- *Throttle-debounce* - programska biblioteka koja računa koliko je prošlo od neke interakcije, pomoću nje je izvedena programska logika za *screensaver* komponentu (“čuvara zaslona”).

3.3.2. Razvojna okolina

Razvojna okolina (eng. *development environment*) predstavlja skup alata i metoda kojima se softver (u ovom slučaju web-aplikacija) postupno razvija, naziva se još i testna okolina jer joj je uloga omogućiti isprobavanje funkcionalnosti i kvalitete rada softvera. U konačnici, po završetku razvoja, softver se kopira u produkcijsku okolinu (eng. *production environment*) koja predstavlja okolinu u kojoj se finalni softver koristi za stvarnu upotrebu (takozvana produkcija).

Razvojna okolina u kojoj se razvijala ova web-aplikacija sastoji se od tri ključna softvera od kojih svaki služi za jedan bitan dio u razvoju web-aplikacije. To su *Git*, *Node.js* i *Visual Studio Code*.

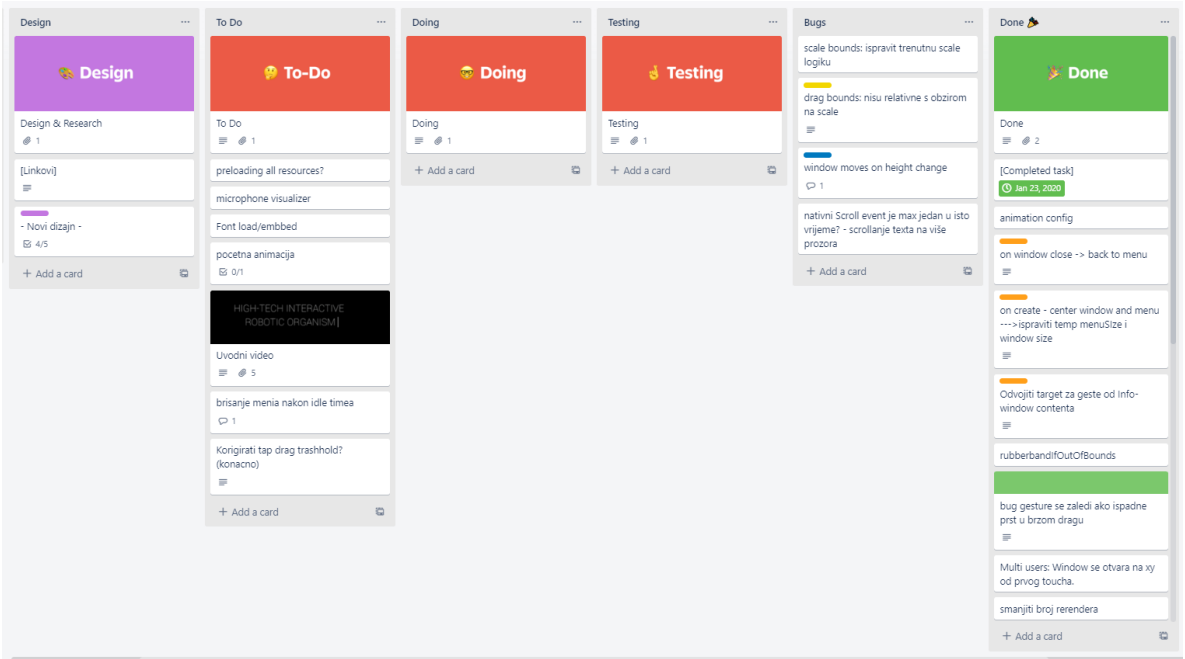
Git je softverski alat koji služi za upravljanje verzijama programskog koda [26]. *Git* svakom projektu omogućava spremanje različitih stanja i verzija programskog koda, te stvara i vodi povijest promjena koda. Pomoću *Git*-a se može pregledati u kojim su datotekama, kad i koje promjene unesene i pohranjene. Također služi kako bi se razvojni programeri mogli vratiti na stariju verziju projekta, na primjer u slučajevima kada postoje problemi i greške na novijoj verziji projekta.

Kako *JavaScript* originalno funkcionira samo unutar korisničkog preglednika razvijen je *Node.js* kao ideja korištenja *JavaScript* programskog jezika van njegove zamišljene svrhe. *Node.js* [27] je softver koji omogućuje korištenje *JavaScript* programskog jezika kao *back-end* rješenja, to jest za pokretanje *JavaScript* procesa na serveru (*JavaScript runtime*). Važan softver koji radi u sklopu *Node.js* okruženja je *npm* (*Node Package Manager*) [28]. *Npm* služi za dohvaćanje (preuzimanje) dodatnih softverskih paketa (biblioteka) koji se kao moduli integriraju u programski kod projekta (Isječak koda 1).

Visual Studio Code (zvan *VS Code*) [29] je primarno uređivač koda (eng. *code editor*), što znači da služi za pisanje i razvoj programskog koda . Vrlo je popularan jer podržava većinu programskih jezika te nudi veliki broj ekstenzija s kojima si razvojni programer prilagođava alate po potrebi. Te ekstenzije, koje su većinom s trećih strana, mogu nadograđivati *VS Code* i do te mjere da on počne funkcionirati ne samo kao uređivač koda već i kao integrirana razvojna okolina (eng. *IDE - Integrated Development Environment).*

U razvojnoj okolini ove web-aplikacije *VS Code* je osim funkcije uređivača koda služio i za upravljanje s integriranim *Git* sustavom, za spremanje promjena u kodu i izradu različitih verzija programskog rješenja te sinkronizaciju između njih. Osim toga, na računalu za razvoj se kroz *VS Code* pokretao *Node.js* server u razvojnom načinu (eng. *development mode*) kako bi se preko lokalne mreža web-aplikacija testirala na interaktivnom stolu.

Za praćenje napretka projekta, upravljanje zadacima i sinkronizaciju s dizajnerima korišten je *Trello*. *Trello* je web servis koji pomaže u upravljanju projekta i kolaboraciji s timom [30]. Vizualno organizirajući informacije i zadatke na principu ploče i kartica (Slika 18), *Trello* praktično prikazuje tko trenutno radi na kojem zadatku i koji se status određenog zadatka.



Slika 18. *Trello* ploča korištena za upravljanje projektom

3.3.3. Razvoj interakcija s gestama dodira

Glavni izazov razvoja ove web-aplikacije bio je tehnički uspješno izvesti zamišljene interakcije s elementima sučelja. Prvi problem je kako u web-aplikaciji, koristeći tehnologije koje web platforma pruža, otkrivati i iskoristiti kompliciranije korisničke geste dodira zaslona osjetljivog na dodir. Drugi problem je, prema zamišljenoj funkcionalnosti web-aplikacije, omogućiti na zajedničkom uređaju (interaktivnom stolu) istovremenu interakciju sa sučeljem za više korisnika.

Za oba problema se iznimno korisnom pokazala programska biblioteka *use-gesture* koja olakšava korištenje interakcijskih događaja dodirom i mišem (eng. *touch and mouse events*) [31]. Osnovne interakcijske događaje prepoznaje i web preglednik bez dodatnih biblioteka, ali *use-gesture* pruža vlastite naprednije funkcije za praćenje interakcijskih gesti na bilo kojem elementu sučelja. Neke od korisnih dodatnih stanja koje pruža su brzina, udaljenost i smjer pokreta geste, istovremeni broj dodira te razlika između pokreta trenutne i prošle geste. Pritom ova biblioteka ostavlja slobodu razvojnom programeru da s obzirom na interakcijsku gestu implementira daljnju programsku logiku po želji.

Funkcije koje *use-gesture* pruža za upravljanje s gestama dodira su *useDrag* za pokret pomicanja prsta*, useScroll* za vertikalno pomicanje sadržaja van ekrana*, usePinch* za pokret „štipanja” s dva prsta i *useGesture* za povezivanje više gesti istovremeno. *Tap* i *swipe* geste se namjerno moraju izvoditi s ponuđenim stanjima unutar *useDrag* funkcije kako zbog same sličnosti pokreta ne bi dolazilo do neželjene kolizije između gesti dodira, povlačenja i pomicanja (*tap*, *swipe*, i *drag* geste*)*.

Uz samu interakciju važna komponenta za kvalitetan doživljaj interakcije je animacija. Zato u kombinaciji s *use-gesture* bibliotekom koristi i *react-spring* biblioteka čija je svrha animirati komponente u sučelju, to jest vrijednosti njenih stilova koje se mijenjaju zahvaljujući određenoj korisničkoj interakciji.

React-spring je biblioteka koja omogućuje animiranje pojedinih elemenata sučelja imitirajući fizička svojstva opruge [32]. Što znači da animacija ne funkcionira na klasičan pristup s definiranom krivuljom i određenim vremenskim trajanjem. Već se, za element koji se animira, vrijednost nekog stilskog atributa računa (metodom interpolacije) pomoću svojstava opruge. Ta svojstva su masa, napetost, trenje, preciznost, brzina i tako dalje... Imitiranjem fizičkih svojstava iz stvarnog svijeta postižu se kontinuirane i prirodnije animacije što direktno poboljšava fluidnost interakcija. Biblioteka sadrži nekoliko predefiniranih postavki animacije, koje predstavljaju balansirane nizove navedenih svojstava opruge. S tim predefiniranim postavkama se može izabrati hoće li animacija, to jest opruga biti nježna, kruta, klimava, usporena ili ljepljiva.

3.3.3.1. Implementacija komponente izbornika

Nakon korisničkog dodira bilo gdje na zaslonu interaktivnog stola pojavljuje se izbornik. Zamišljena interakcija s izbornikom je ta da korisnik može gestom pomicanja (*drag*) proizvoljno pomicati izbornik unutar granica zaslona te gestom jednog dodira (*tap*) birati opcije izbornika kako bi otvorio odabrani sadržaj u komponenti prozora ili kako bi zatvorio izbornik. Uz svaku interakciju pridružena je i odgovarajuća animacija, to su animacije pojavljivanja, pomicanja i zatvaranja izbornika.

Sljedeća četiri isječka koda (2, 3, 4 i 5) u dijelovima prikazuju kako su programskom logikom razvijene zamišljene interakcije s izbornikom te pripadajuće animacije. U izvornom kodu sva četiri isječka pripadaju istoj komponenti izbornika te ih je kao takve potrebno promatrati jer sadrže i dijele iste programske reference.

Isječak koda 2 u prvom bloku prikazuje definiranje `useSpring` funkcije animacije koja se sastoji od dva dijela, `menuStyle` koji predstavlja stilske vrijednosti komponente izbornika i `animate` poziv koji će se koristiti za animiranje promjene stila komponente. Također pri definiranju `useSpring` funkcije animacije zadaje se ulazna animacija, to jest ona koja predstavlja tranziciju pri prvom pojavljivanju izbornika (linije koda #2 i #3). Ulazna animacija se svodi na prijelaz neprozirnosti i veličine izbornika (`opacity i scale`) s vrijednosti 0 na 1 (vrijednosti predstavljaju raspon 0% – 100%), a izbornik se pojavljuje na poziciji koja je dotaknuta na zaslonu.

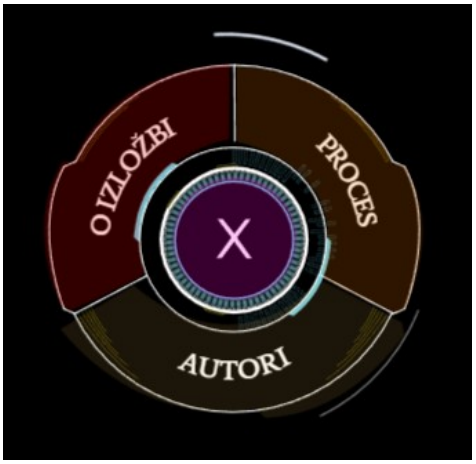
U drugom bloku (Isječak koda 2) se nalazi `return` funkcija koja prikazuje komponentu izbornika. Komponenta izbornika (`MenuCSS`) ima dva zadana atributa (linija koda #*8*). Prvi je `ref` koji predstavlja referencu za nalaženje komponente izbornika u DOM-u kako bi se na tu komponentu pridružile funkcije gesti. A drugi je `styles` koji predstavlja skup stilova komponente izbornika, to jest stilova koji će se mijenjati pomoću `useSpring` funkcije animacije. Komponenta izbornika (`MenuCSS`) je sastavljena od pet elemenata, prva se

prikazuje komponenta koja sadrži SVG ilustraciju izbornika, a za njom četiri elementa koji služe kao gumbi (Slika 19). Ta četiri elementa (*linije koda #10 – #13*) su transparenta maska (preko SVG ilustracije izbornika), njihova svrha je proslijediti informaciju (`data-menu`) koji je točno element pritisnut kada dođe do korisničke interakcije s izbornikom.

Isječak koda 2. Inicijalizacija animacije i prikazivanje izbornika

```
const [menuStyle, animate] = useSpring(() => ({
  from: { opacity: 0, scale: 0, x: start_x, y: start_y },
  to: { opacity: 1, scale: 1, x: start_x, y: start_y },
  config: config.wobbly,
}));

return (
 <MenuCSS ref={menuDOM_target} style={menuStyle}>
   <MenuSVG />
    <div className="menu-button button-info" data-menu={1} />
    <div className="menu-button button-gallery" data-menu={2} />
    <div className="menu-button button-map" data-menu={3} />
    <div className="closeMenu" data-menu={-1} />
  </MenuCSS>
 );
```



Slika 19. Četiri opcije unutar izbornika (elementi označeni bojom)

Isječak koda 3. Programska logika *tap* i *drag* geste za izbornik

```
1  useDrag(
2    ({ event, active, offset: [x, y], tap }) => {
3      if (tap) {
4        let menuChoice =
             parseInt(event.target.getAttribute("data-menu"));
5        if (!menuChoice) return;
6        if (menuChoice === -1) removeMenu();
7        else props.menuClick(event, menuChoice);
8      } else {
9        animate.start({
10          x: x,
11          y: y,
12          scale: active ? 0.8 : 1,
13        });
14     }
15   },
16   { (...) },
17 );
```

Isječak koda 3 prikazuje implementaciju `useDrag` funkcije i razvoj programske logike pomicanja izbornika (*drag* geste) uz koju je uključena i logika za dodire opcija izbornika (*tap* gesta).

Linija koda #2 prikazuje koje će se informacije o gesti koristiti za razvoj logike. Sama logika se sastoji od dva dijela (bloka koda), to jest od jedne *if-else* naredbe. Prvi blok (linije koda #4 – #7) se izvršava ako je `useDrag` funkcija vratila informaciju da započeta gesta pomicanja izbornika ustvari nije gesta pomicanja nego je korisnik samo napravio jedan dodir (*tap* gestu). U slučaju da je korisnik napravio dodir na izbornik, prvo se provjerava koju je opciju izbornika (gumb) dotaknuo (Slika 19), zatim:

1. ako nije dotaknut ni jedan dio izbornika (gumb) s ponuđenom opcijom – funkcija se prekida

2. ako je dotaknuta opcija za zatvaranje izbornika – pokreće se funkcija zatvaranja izbornika i funkcija završava
3. pokreće se funkcija za otvaranje odgovarajućeg prozora s obzirom na dotaknutu opciju izbornika i funkcija završava

U slučaju da korisnik nije napravio samo jedan dodir na izborniku već ga duže drži ili pomiče (*drag* gesta), umjesto prvog bloka `if-else` naredbe izvršava se drugi koji se sastoji samo od funkcije pokretanja animacije (linije koda #9 – #13). Izbornik se animira tako da mu se trenutna pozicija (`x` i `y` atributi) mijenja na pomak prsta, to jest na svaku poziciju za vrijeme geste (*drag*) pomicanja. Drugi dio animacije (linije koda #12) mijenja veličinu (`scale`) izbornika na 80% dokle god korisnik pomiče izbornik, to jest dok je gesta aktivna (`active`), jednom kad gesta završi veličina izbornika se vraća na originalnu veličinu (100%).

Isječak koda 4. Korištene opcije *useDrag* geste za izbornik

```
1  useDrag(
2   ({ event, active, offset: [x, y], tap }) => {
3     (...)
4   },
5   {
6     target: menuDOM_target,
7     from: () => [menuStyle.x.get(), menuStyle.y.get()],
8     bounds: { left: 0, right: width, top: 0, bottom: height},
9     rubberband: 0.5,
10    filterTaps: true,
11  },
12 );
```

Isječak koda 4 prikazuje implementaciju postavki `useDrag` funkcije, to jest korištene opcije kako bi se preciznije podesila funkcionalnost geste. Funkcija `useDrag` se zajedno s logikom geste (iz Isječka koda 3) pridružuje komponenti izbornika (u Isječku koda 2) pomoću `target` opcije s kojom se povezuje `menuDOM_target` referenca komponente izbornika (linija koda #6).

- Opcija `from` definira s koje pozicije će započinjati animacija pomaka izbornika, u ovom slučaju izborniku će pri animaciji pomicanja početna pozicija uvijek biti ona trenutna, to jest neće se vraćati na neku originalnu poziciju već će ostati na svojoj posljednjoj poziciji po završetku geste pomicanja (linija koda #7).

- Opcija `bounds` definira granice okvira unutar kojega se zadani element može pomicati. U slučaju ove web-aplikacije, površina po kojoj se izbornik smije kretati je cijeli zaslon, zato zadane koordinate granica po širini: 0 – širina prozora (cijeli zaslon), odnosno po visini: 0 – visina prozora (linija koda #8). Da ove opcije nema korisniku bi bilo omogućeno neželjeno ponašanje, mogao bi izbornike pomaknuti van granica zaslona (dokle god je vidljiv barem dio izbornika kako bi ga se moglo doticati).

- Opcija `rubberband` se nadovezuje na opciju `bounds` definirajući fleksibilnost granica okvira koje je zadala opcija `bounds`. Vrijednost ove opcije predstavlja koeficijent elastičnosti koji dopušta elementu djelomični prelazak granica okvira te ga vraća unutar granica okvira čim gesta pomicanja završi.

- Opcija `filterTaps` zadaje funkciji geste da provjerava je li korisnička interakcija stvarno bila gesta pomicanja (*drag*) ili je korisnik samo dotaknuo element (*tap* gesta). U slučaju da je nakon interakcije dodirom ukupni pomak dodira manji od 3 pixela gesta će smatrati da se radi o jednom dodiru, a ne o pomicanju. Granica ukupnog pomaka se također može proizvoljno podesiti. Ova opcija se koristi jer izbornik zahtjeva različitu reakciju s obzirom o kojoj se točno gesti (programska logika iz Isječka koda 2).

Isječak koda 5. Funkcija zatvaranja i uklanjanja izbornika

```
1  const removeMenu = () => {
2   animate.start({
3     scale: 0,
4     opacity: 0,
5     onRest: () => {
6       props.removeFromArr();
7     },
8   });
9 };
```

Isječak koda 5 prikazuje funkciju koja se poziva u Isječku koda 3 kada se dotakne opcija za zatvaranje izbornika. Ove funkcije pokreće animaciju veličine i neprozirnosti izbornika (`scale` i `opacity`) na vrijednost nula. Kada se animacija završi, izbornik se uklanja s popisa postojećih izbornika. (linija koda #6).

3.3.3.2. Implementacija komponente prozora

Komponenta prozora se pojavljuje nakon što korisnik odabere jednu od tri opcije u izborniku gdje svaka od opcija predstavlja jednu vrstu prozora, to jest jednu od kategorija sadržaja (sadržaj o izložbi, o autorima i galerija). S obzirom na različit sadržaj, svaka vrsta prozora ima drugačiji raspored sadržaja stoga se i oblik samog prozora prilagođava tom sadržaju. Sadržaj prozora je podijeljen u logične cjeline te se unutar jednog prozora može nalaziti više „stranica" sadržaja koje se prikazuju po uzoru na „tabove" u internet pregledniku. „Tabovi" funkcioniraju kao gumbi pomoću kojih se korisnik (*tap* gestom) navigira kroz sadržaj unutar jednog prozora. Druga zamišljena korisnička interakcija jest manipulacija s otvorenim prozorom. Osim mijenjanja pozicije prozora pomoću geste pomicanja (*drag* gesta), korisnik može povećavati i smanjivati te rotirati prozor koristeći gestu „štipanja" (*pinch* gesta).

Implementacija gesti koje se izvode s jednim prstom, pomicanje i jedan dodir (*drag* i *tap geste),* približno je ista kao i kod komponente izbornika. No za promjenu veličine i rotaciju prozora potrebno je implementirati funkciju za još jednu gestu, gestu „štipanja" (*pinch)* koja se izvodi s dva prsta. Kako bi se obje funkcije gesti povezale na istu komponentu prozora koristi se `useGesture` funkcija koja će grupirati dvije te funkcije geste. Umjesto pozivanja dvije funkcije za geste (`useDrag` i `usePinch`) koristi se `useGesture` funkcija u kojoj se ove dvije geste navode kao događaji `onDrag` i `onPinch` (Isječak koda 6), a ostala funkcionalnost je ista kao u primjeru komponente izbornika.

Isječak koda 6 u linijama koda #3 – #6 prikazuje samo dio `onDrag` događaja (geste), cijela logike geste nije prikazana u isječku koda jer se sastoji od iste logike kao i u slučaju komponente izbornika. Dio te logike koji se razlikuje je linija koda #5, ona služi kao dodatna provjerava je li se za vrijeme *drag* geste neželjeno događa i druga gesta (`pinching`), ako da *drag* gesta je prekida. Ako

ne bi bilo navedene provjere postoji mogućnost neželjenog scenarija u kojem korisnik izvodi drag gestu jednim prstom, a zatim doda drugi prst kako bi započeo *pinch* gestu miješajući tako dvije različite geste istovremeno.

U Isječku koda 6, linije koda #8 – #27 prikazuju kako je u dva koraka implementirana gesta „štipanja" (*pinch gesta*) kojom će se manipulirati veličina, rotacija i pozicija prozora. Prvi korak je blok koda if naredbe, pomoću `first` vrijednost koju `onPinch` funkcija vraća, provjerava se radi li se o samom početku (prvom dodiru) geste. Ako je gesta tek započeta, pomoću `memo` funkcije se pohranjuju početne vrijednosti koordinata pozicije i središta prozora (linija koda #14). Početne vrijednosti služe za drugi korak, a to je računanje novih koordinata pozicije prozora koje će biti relativne s obzirom na početne vrijednosti i trenutni pomak geste s dva prsta (`movement: [ms]`) (linije koda #16 i 17). Zatim se animiraju atributi prozora sa svim promijenjenim vrijednostima, veličina (`scale`) koja postaje relativna razlika veličine (`s`) s obzirom na početnu veličinu, rotacija (`rotateZ`) koja postaje nova vrijednost kuta zakrivljenosti (`a`) koju gesta vraća, te koordinate pozicija izračunate u prošlom koraku (`x` i `y`). Boja obruba (`borderColor`) se mijenja dok je gesta aktivna te se vraća u početnu boju po završetku geste.

Kao što je bio slučaj kod implementacije komponente izbornika tako i ova komponenta (prozor) treba imati ograničenja u korisničkoj manipulaciji koju dozvoljava. Prozor na isti način postavlja granice za gestu pomicanja unutar kojih se prozor može kretati. Dodatne granice koje prozor postavlja su one za promjenu veličine kako korisnik ne bi mogao previše smanjiti ili povećati prozor (ispod donje, odnosno iznad gorenje granice) jer bi na taj način ta komponenta izgubila svoju funkcionalnost. Također je moguće i ograničiti kut zakrivljenosti prozora kako ga korisnik ne bi mogao rotirati izvan predviđenih granica. No ta se opcija u ovom slučaju ne koristi jer je zamišljeno da se prozor može rotirati puni krug kako bi si ga korisnik mogao prilagoditi bez obzira na kojoj se srani zaslona, odnosno interaktivnog stola korisnik nalazi.

Isječak koda 6. Implementacija *pinch* geste za komponentu prozora

```
useGesture(
 {
  onDrag: ({ event, tap, active, offset: [x, y], pinching, cancel
   }) => {
   (...)
   if (pinching) return cancel();
  },

  onPinch: ({ active, first, movement: [ms], offset: [s, a],
              origin: [ox, oy], memo
  }) => {
    if (first) {
     const { width, height, x, y } =
            domTarget.current.getBoundingClientRect();
     const tx = ox - (x + width / 2);
     const ty = oy - (y + height / 2);
     memo = [windowStyle.x.get(), windowStyle.y.get(), tx, ty];
    }
    const x = memo[0] - (ms - 1) * memo[2];
    const y = memo[1] - (ms - 1) * memo[3];
    api.start({
     scale: s * theme.const.window_startScale,
     rotateZ: a,
     x: x,
     y: y,
     borderColor: active ? pinchColor : "black",
    });
    return memo;
   },
  },
  {
   (...)
  },
);
```

3.3.4. Implementacija dizajn sustava

Dobiveni dizajn sustav oblikovan u *Figmi* razvijen je u programski kod sučelja web-aplikacije pomoću *styled-components* biblioteke. *Styled-components* je programska biblioteka za *React* koja omogućuje razvoj CSS stilova unutar JSX sintakse *React*-a [33], a funkcionira što se posebno definira svaki element sučelja kao *React* komponenta kojoj je pridružen stil pisan s klasičnom CSS sintaksom. Tako razvijene komponente se mogu višekratno koristiti na različitim mjestima unutar web-aplikacije zajedno s njihovim stilovima (dizajnom). To znači da se dizajn sučelja izrađen po principu „atomskog dizajna" razvija u sučelje za web-aplikaciju isto po principu „atomskog dizajna", gradeći kompleksnije komponente od jednostavnijih.

Sljedeći isječci koda prikazuju razvoj komponenti sa stilovima primjenom *styled-components* biblioteke te primjenu tih komponenti za prikazivanje u korisničkom sučelju. Isječak koda 7 prikazuje na koji način se razvijena komponenta obavijesti prikazuje u korisničkom sučelju.

A Isječak koda 8 prikazuje definiranje stilova komponente za obavijest.

Isječak koda 7. Prikazivanje komponente obavijesti u sučelju

```
<MessageCSS>
   <div className="text">
     <Text_Description>
       {Messages.menuLimit} {MENU_LIMIT}
     </Text_Description>
   </div>
   <div className="line" />
</MessageCSS>
```

Isječak koda 8. Definiranje stila komponente obavijesti

```
import styled from "styled-components";

export const MessageCSS = styled(animated.div)`
  will-change: transform, opacity;
  z-index: 999;
  position: absolute;
  padding: 12px;
  background-image: url("graphics/message.svg");
  background-repeat: no-repeat;
  background-size: contain;
  .line {
      margin-top: -2px;
      padding: 0px 16px 24px 16px;
      content: url("graphics/message-line.svg");
  }
  .text {
      max-width: 43ch;
      padding: 26px 102px 0px 24px;
  }
`;
```

Isječak koda 9 prikazuje definiranje komponenata sa stilovima tipografije, komponente za naslov i podnaslov.

A isječak koda 10 prikazuje primjenu tih i ostalih komponenti u komponenti prozora te kako je pomoću manjih komponenti izrađena i prikazana komponenta prozora u korisničkom sučelju.

Isječak koda 9. Definiranje stilova komponenti naslova i podnaslova

```
import styled from "styled-components";

export const Text_Title = styled.span`
  /* Naslov1 */
  font-family: Avenir Next Condensed;
  font-style: normal;
  font-weight: 600;
  font-size: 2.5rem;
  line-height: 3rem;
   letter-spacing: 0.035em;
`;

export const Text_Subtitle = styled.span`
  /* Podnaslov */
  font-family: Avenir Next Condensed;
  font-style: normal;
  font-weight: 500;
  font-size: 1.5rem;
  line-height: 2rem;
  letter-spacing: 0.035em;
`;
```

Isječak koda 10. Prikazivanje komponente prozora pomoću manjih komponenti

```
<MainWindowCSS ref={ref} style={windowStyle}>
  <Header>
    <Image src="logo-grf.svg" height={64} width={64} />
    <Text_Title> O IZLOŽBI </Text_Title>
    <ArrowsButton />
    <CloseButton />
  </Header>
  <BottomLine />
  <PageMenu>
    <Text_Subtitle> HoloGRF </Text_Subtitle>
    <Text_Subtitle> Vrste holograma </Text_Subtitle>
  </PageMenu>

   (... sadržaj prozora ...)

   <Footer>
     <Image src="qr-instagram.svg" height={150} width={150} />
     <Text_Body_secondary> {InfoText.footer_qr} </Text_Body2>
     <Text_Body> {InfoText.footer_title} </Text_Body>
   </Footer>
</MainWindowCSS>
```

3.3.5. Otklanjanje grešaka, testiranje i iteracije

Kako je ova web-aplikacija razvijana za nekonvencionalni uređaj (interaktivni stol), nije bilo dovoljno razvijati programsko rješenje na lokalnom računalu, već je bilo potrebno svaku funkcionalnost intenzivno testirati na samom interaktivnom stolu. A to je ujedno bio i jedini uređaj na kojem se moglo vršiti testiranje razvijenih rješenja jer druge vrste uređaja ili nemaju zaslon osjetljiv na dodir ili ne podržavaju toliki broj dodirnih interakcija (zbog manjih dimenzija zaslona). Također, iako web kao platforma podržava tehničke zahtjeve zamišljene web-aplikacije, svejedno takav slučaj upotrebe web tehnologija nije uobičajen i stoga je bilo potrebno tražiti dodatna rješenja za nastale poteškoće u razvoju. U nastavku su navedeni neki od problema koji su se pokazali pri razvoju.

Na primjer, zbog velike osjetljivosti, to jest rezolucije zaslona interaktivnog stola nije se svaki put iz prvog pokušaja moglo aktivirati gumbe u sučelju. Gumb bi reagirao samo ako je gesta dodira bila izrazito precizna i bez dodatnih pomaka. Čim bi korisnik djelomično pomaknuo prst pri dodiru gumba ili dodirnuo gumb sa širom površinom prsta – web-aplikacija bi registrirala da se radi o gesti pomicanja (*drag*) umjesto o jednom dodiru (*tap*). Rješenje problema je bilo povećati prag tolerancije za gestu dodira kako bi bio potreban veći pomak da bi ga web-aplikacija registrirala kao gestu pomicanja.

Jedan od problema s interakcijom više korisnika istovremeno bio je taj što elementi sučelja, s kojima korisnici manipuliraju, nemaju implementiranu međusobnu koliziju (kako je i zamišljeno). Zbog toga je mogući scenarij u kojem prvi korisnik ima otvoren jedan prozor, a drugi korisnik otvaranjem svog prozora neposredno blizu prvog prozora prekriva taj prvi prozor te prvi korisnik više ne može doći do njega. Implementirano je rješenje koje zadnji dotaknuti element sučelja premješta na vrh zaslona. Tako korisnik može vratiti u fokus prozor koji se našao ispod nekog drugog te ih manipulacijom organizirati po površini zaslona.

Jedna od limitacija web tehnologije pokazala se funkcionalnost „trake za pomicanje” po stranici (*scroll)*. Prvi pokušaji razvoja komponente prozora uključivali su *scrollbar* traku za vertikalno pomicanje kao uobičajeno rješenje za otkrivanje dodatnog sadržaja koji ne stane unutar jedne površine prozora. Međutim pokazalo se kako je *scroll* pomak jedina od potrebnih interakcija koja ne funkcionira na više prozora istovremeno.

Kako korisnik dok se služi interaktivnim stolom ne bi morao razmišljati ili biti svjestan o kakvoj se točno aplikaciji radi, bilo je potrebno pobrinuti se da uređaj i softver na njemu ne pokazuju nikakve suvišne informacije koje bi omele korisničko iskustvo. Na primjer interaktivni stol ima zadane postavke za imitiranje funkcija miša interakcijama na dodir. Tako da je bilo potrebno ukloniti funkcionalnost koja ako se zaslon dodirne s dva prsta istovremeno, otvorit će se izbornik s opcijama softvera, onaj izbornik koji se inače otvara s desnim klikom miša. Uz navedene zahvate i pokretanje web-aplikacije u punom zaslonu (*fullscreen*) korisnik bi bio potpuno uronjen u interakciju s web-aplikacijom.

Web-aplikacija kao takva nema ograničenja ni razloge ograničavati broj komponenti i animacija koje se istovremeno odvijaju na zaslonu. Ipak, ograničen je broj izbornika (to jest prozora) koji istovremeno mogu biti otvoreni na šest komada što ujedno predstavlja maksimalan broj korisnika koji fizički mogu biti za interaktivnim stolom istovremeno. S tim ograničenjem spriječen je scenarij u kojem korisnik (ili više njih) mogu (u teoriji) beskonačno otvarati nove izbornike i prozore te time prigušiti i interaktivnu površinu zaslona i samu funkcionalnost web-aplikacije. Ograničenje je implementirano na takav način da nakon što je šest izbornika (ili prozora) otvoreno, na svaki sljedeći pokušaj otvaranja novog izbornika na nekoliko sekundi se pojavljuje prozor s obavijesti kako je otvoren maksimalan broj prozora. S ovim ograničenjem je također preventirana potreba za dodatnim optimiziranjem materijala kao što su grafike komponenti i animacije, koje bi mogle predstavljati problem da se navedene komponente moraju prikazivati deseterostruko više puta nego za što su optimizirane.

3.4. Povratne informacije i korisnička iskustva

Razvijena web-aplikacija je na interaktivnom stolu bila izložena u sklopu HoloGRF - Mrak izložbe na Grafičkom fakultetu. Interaktivni stol je bio postavljen tako da posjetitelji izložbe nakon prolaska polovice izloženih radova s vodičem dolaze do interaktivnog stola te se samostalno pokušaju služiti njime. Svi posjetitelji su bez posebnih poteškoća, u minimalno kratkom ili nikakvom roku savladali načine interakcije s uređajem. Odnosno svima su bile poznate osnovne geste dodira za zaslone osjetljive na dodir te nije bilo naručite krivulje učenja što se tiče uspravljanja i služena web-aplikacijom na interaktivnom stolu.

Manjem dijelu posjetitelja nije na prvi pogled bilo jasno kako započeti interakciju, to jest da je za interakciju sa stolom potreban njihov dodir te da je uređaj ispred njih sa zaslonom osjetljivim na dodir. Razlog tome mogu biti uobičajena pravila ponašanja na izložbi, ili to što je prostorija bila zamračena pa posjetitelji nisu bili sigurni što se točno nalazi ispred njih te nemaju prijašnje iskustvo kako bi prepoznati tu vrstu uređaja, pogotovo jer se češće koristi u vertikalnom položaju dok je ovaj bio u horizontalnom. No sigurno je ključnu ulogu igrao i početni zaslon, stanje web-aplikacije kada ju nitko ne koristi, odnosno „poziv na akciju” (eng. CTA - *call to action*) koji bi kao element u sučelju u obliku teksta ili slike nagovještavao što učiniti i ohrabrivao posjetitelje da započnu interakciju.

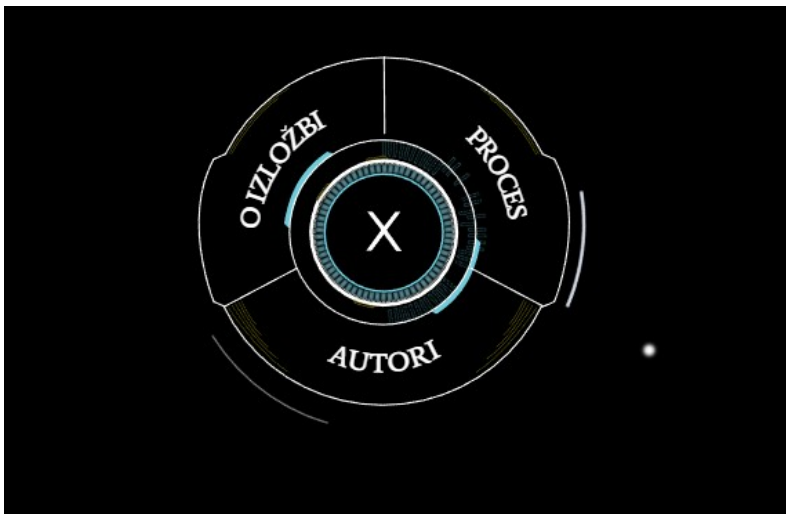
Dio posjetitelja koji je pred interaktivnim stolom pretpostavio da se radi o zaslonu osjetljivom na dodir, je bez poteškoća započeo interakciju s web-aplikacijom te uspješno otkrivao sadržaj o izložbi. Posjetiteljima nije bilo poznato da imaju neke mogućnosti kao što su pomicanje izbornika po zaslonu, što je razumljivo jer su se svi posjetitelji prvi put služili web-aplikacijom (eng. *first-time users*). Korisničko iskustvo bi se u ovom slučaju moglo unaprijediti dodavanjem suptilnih uputstava koja bi korisnicima na nenametljiv način pružala prijedloge što u kojem trenutku mogu napraviti.

Ukupni dojam posjetitelja za interaktivnim stolom se činio iznimno dobrim, posjetitelji su bili zadovoljni iskustvom i iznenađeni fluidnošću interakcije sa stolom zahvaljujući brzini i robusnosti web-aplikacije. Neki posjetitelju su bili čak do te mjere zabavljeni kvalitetno animiranom interakcijom pri otvaranju, zatvaranju i premještanju elemenata po zaslonu da ih nije dodatno interesirao i informativni sadržaj unutar tih elemenata.

Može se sa sigurnošću reći kako posjetitelje ugodno iznenadio interaktivni stol kojeg je ova web-aplikacija na uspješno iskoristila kao platformu. Oni koje je zanimala tehnologija izrade nisu očekivali kako se radi o web-aplikaciji s obzirom da ju je više posjetitelja istovremeno moglo koristiti što je uobičajena namjena te je web-aplikacija bila prezentirana tako da ne otkriva sučelje softvera koji ju pokreće. Posjetitelje je zanimalo koje su još mogućnosti korištenih tehnologija i daljnji planovi razvoja. Te su česta bila pitanja i prijedlozi o izradi video igre, što je očekivano s obzirom na prezentirane interakcije i animacije te dovoljno veliku površinu zaslona koji može podržati više korisnika istovremeno.

3.4.1. Razvijeno sučelje web-aplikacije

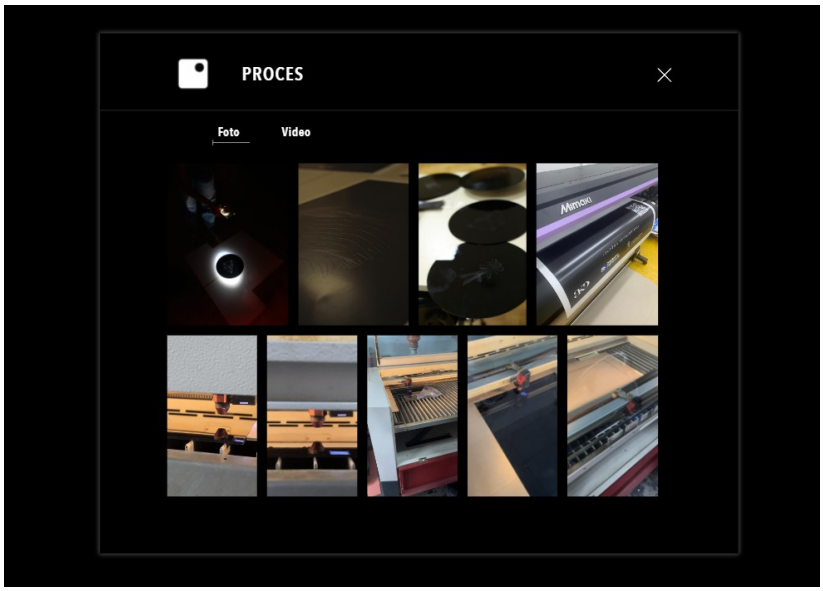
Sljedeće slike su slike zaslona razvijene web-aplikacije, a prezentiraju komponente i korisničkog sučelja.



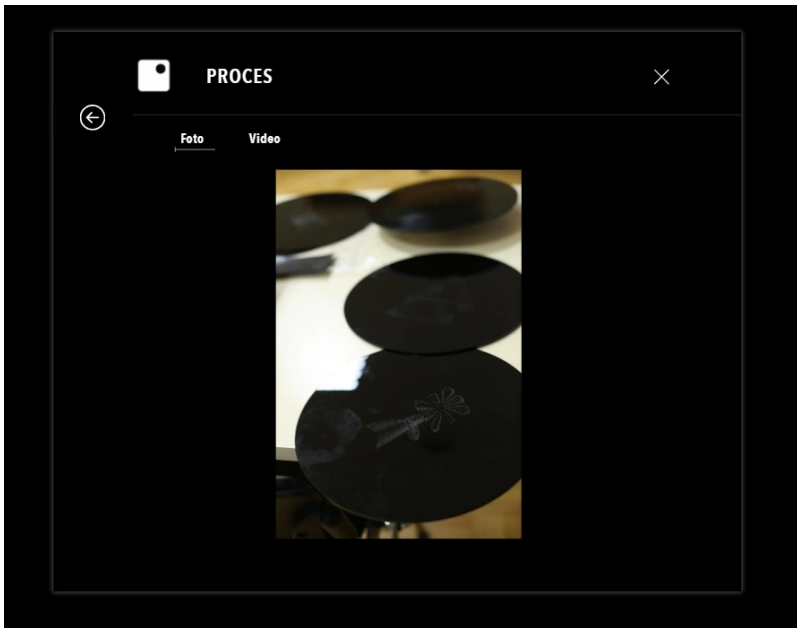
Slika 20. Izbornik (sučelje web-aplikacije)

Slika 21. Obavijest (sučelje web-aplikacije)

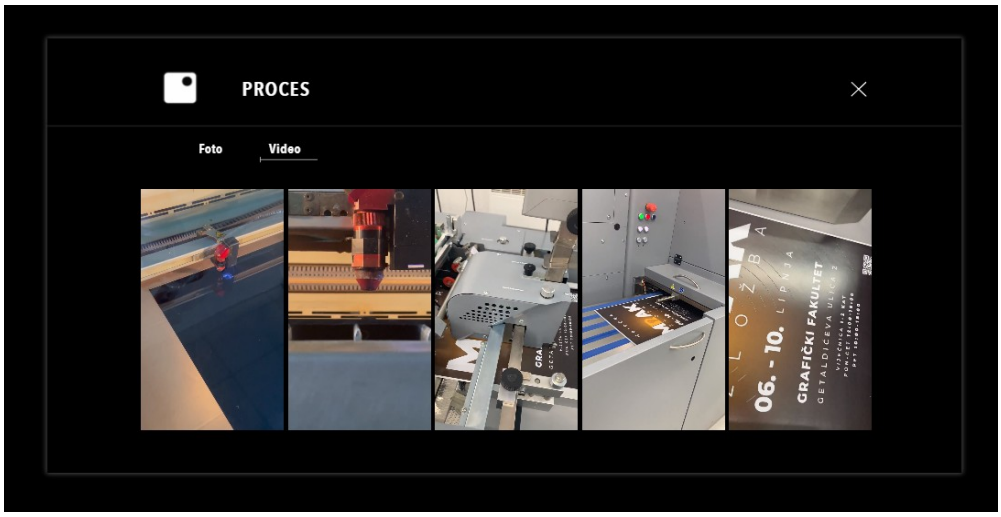
Slika 22. Prozor „O izložbi” (sučelje web-aplikacije)



Slika 23. Prozor „Proces”, galerija (sučelje web-aplikacije)

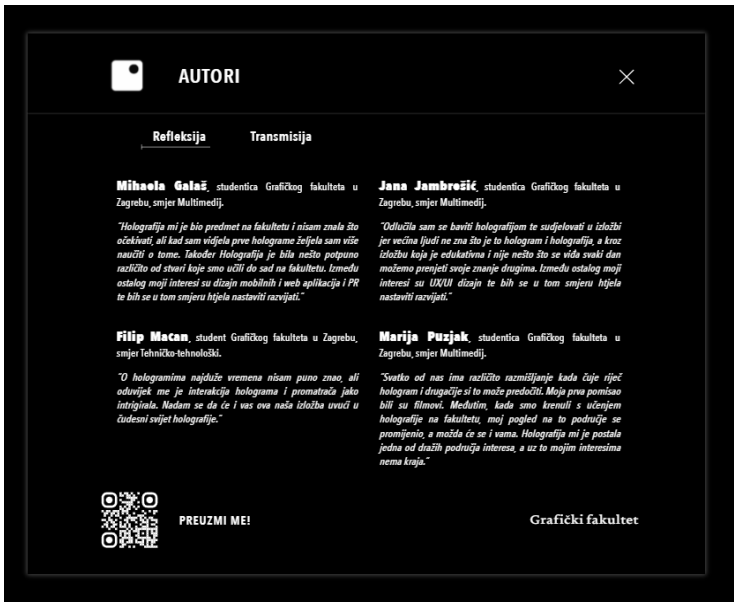


Slika 24. Prozor „Proces”, fotografija (sučelje web-aplikacije)

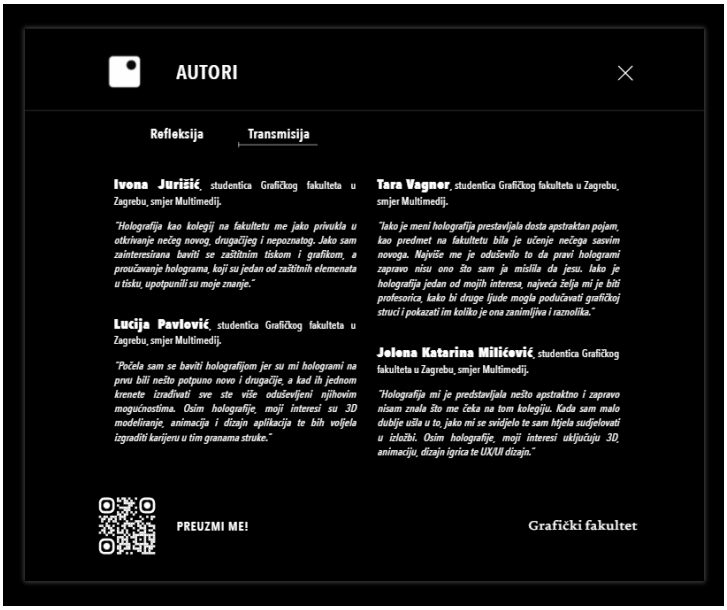


Slika 25. Prozor „Proces”, video (sučelje web-aplikacije)

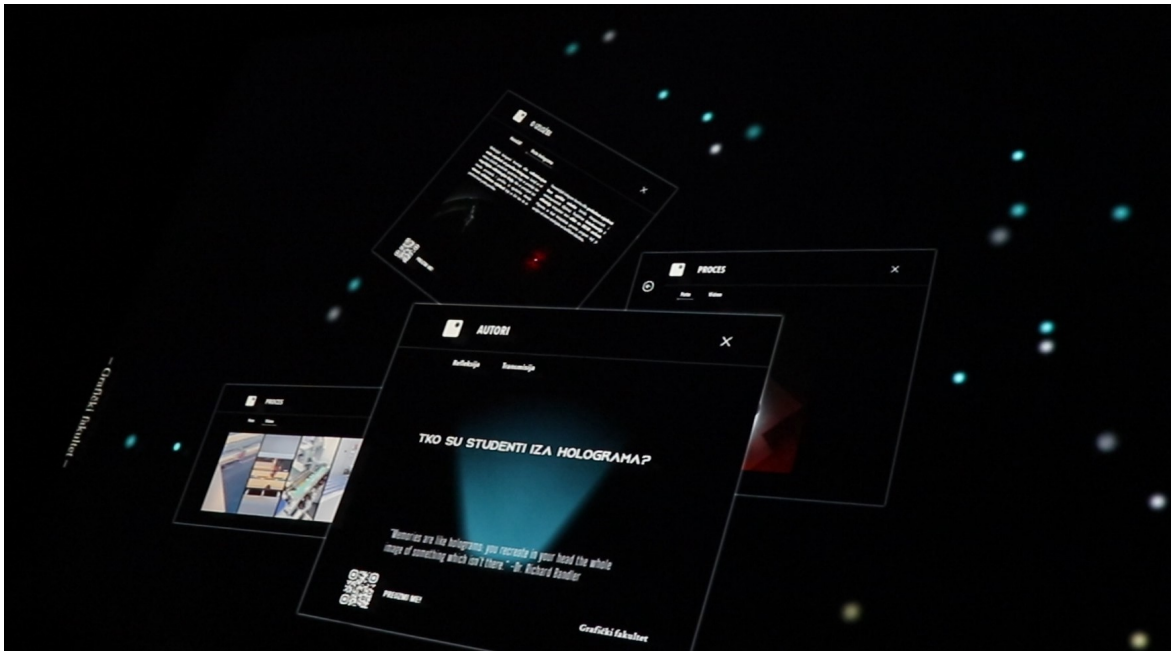
Slika 26. Prozor „Autori”, uvod (sučelje web-aplikacije)



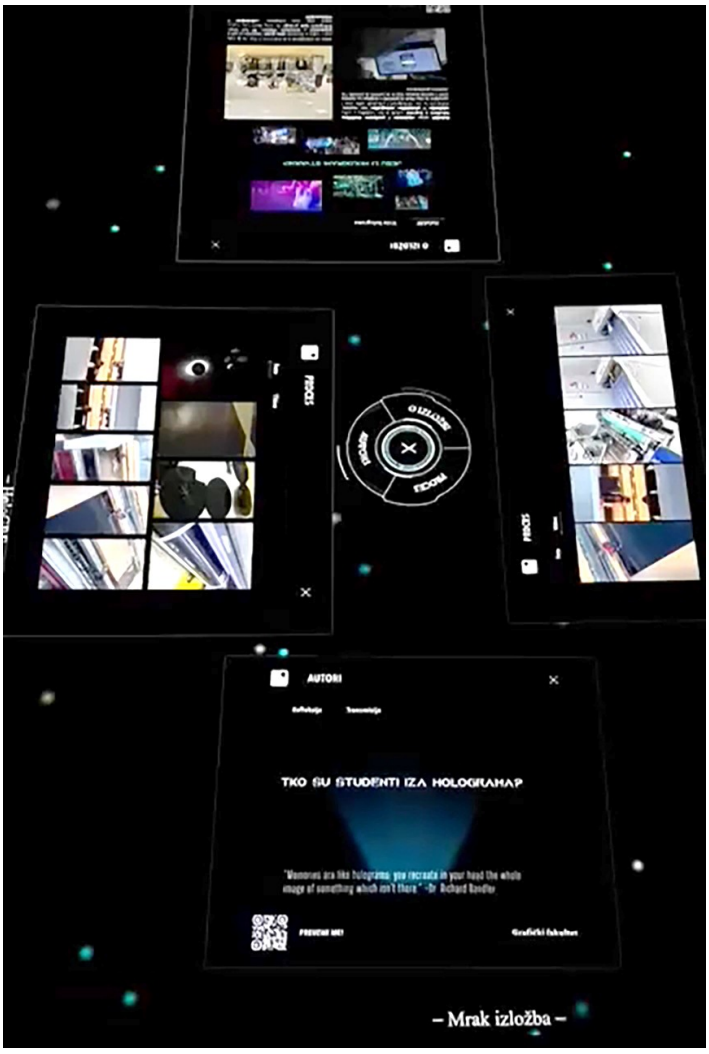
Slika 27. Prozor „Autori”, refleksija (sučelje web-aplikacije)



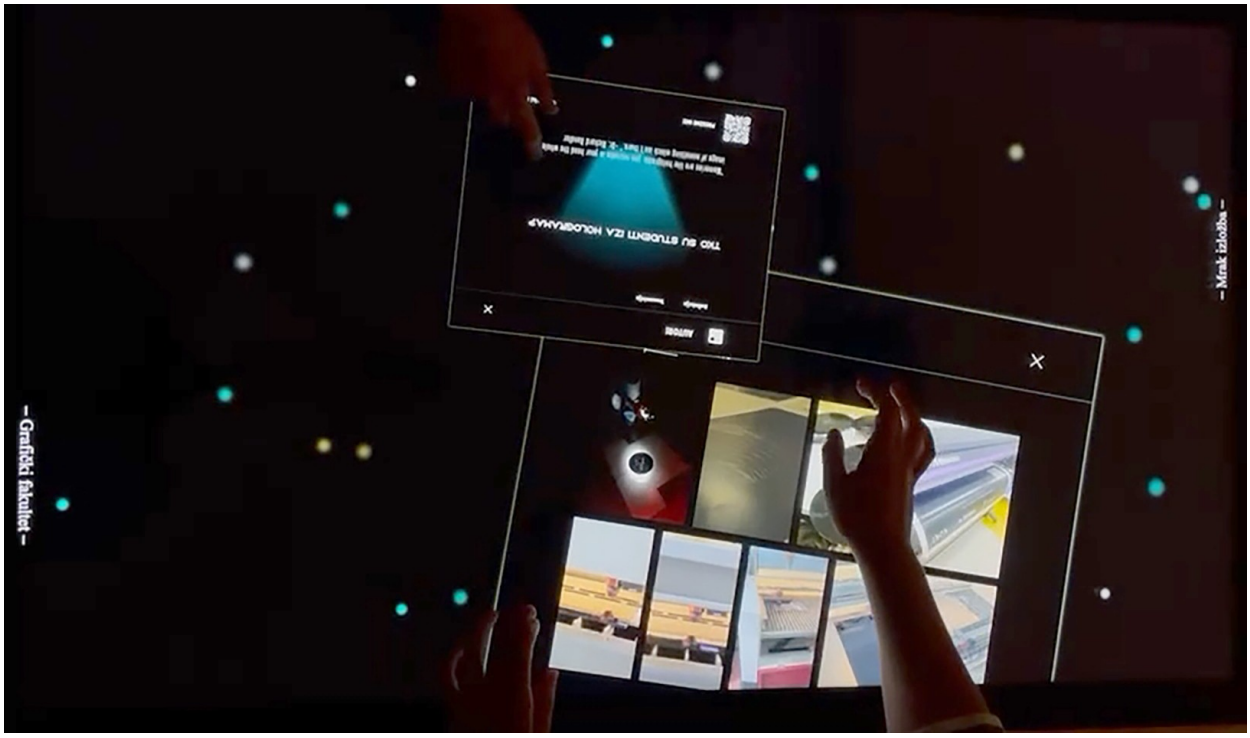
Slika 28. Prozor „Autori”, transmisija (sučelje web-aplikacije)



Slika 29. Fotografija otvorenih prozora na interaktivnom stolu



Slika 30. Fotografija površine interaktivnog stola s otvorenim prozorima



Slika 31. Fotografija istovremene interakcije korisnika s interaktivnim stolom

4. ZAKLJUČAK

Razvijanjem ove interaktivne web-aplikacije potvrđen je značaj dizajna interakcija te važnost primjene utvrđenih principa dizajna interakcija, u svrhu prilagođavanja aplikacije za kvalitetno korisničko iskustvo i upotrebljivost. Što u konačnici rezultira kvalitetnom odnosno uspješnom aplikacijom, potvrđujući princip izrade dizajna usmjerenog na krajnjeg korisnika. Također, potvrđen je način dizajniranja interakcija bazirajući se na utvrđene obrasce kako bi korisniku predviđene interakcije bile poznate i intuitivne bez da iziskuju značajni napor od korisnika. U konačnici, web-aplikacijom je testirana i ključna uloga korisničkog sučelja i pripadajućih animacija na doživljaj korisnika.

Utvrđeno je kako web kao platforma i tehnologija ima nezanemarivu ulogu na spektru tehnologija za razvoj korisničkih aplikacija. Te kako fleksibilnost i prednosti web tehnologija za izradu aplikacija nisu beznačajne, u usporedbi kako s mobilnim tako i sa svim ostalim platformama, što moderni programski alati i rješenja za web platformu potvrđuju.

Samim programskim razvojem ove nekonvencionalne web-aplikacije upravljane gestama dodira, a pomoću *React* programske biblioteke, nije dokazana samo kompetencija standardne web tehnologije, već i velika uloga svestranosti i prilagodljivosti *React* programske biblioteke. *React* se kao vodeća web tehnologija pozicionirao tako da će biti relevantan još duži niz godina, pogotovo ako se uzmu u obzir zastarjele web tehnologije koje se i dalje koriste iako već dugo nisu relevantne u svijetu modernog web sadržaja.

5. LITERATURA

[1] H. Sharp, J. Preece, Y. Rogers (2019), Interaction Design: Beyond Human-Computer Interaction, 5th Edition, Wiley, Indianapolis, IN

[2] J. Tidwell, C. Brewer, A. Valencia (2020), Designing Interfaces: Patterns for Effective Interaction Design, 3rd Edition, O'Reilly Media, , Sebastopol, CA

[3] S. Krug (2014), Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability, 3rd Edition, New Riders,

[4] Fingers, Thumbs & People - Steven Hoober, https://www.slideshare.net/shoobe01/fingers-thumbs-people-15-minute-version (pristupljeno: 10. 8. 2022.)

[5] Google: “Responsive Web Design Now Even More Important”, https://arcainteractive.com/google-responsive-website-design-now-even-important/ (pristupljeno: 10. 8. 2022.)

[6] States - Material Design, https://material.io/design/interaction/states.html (pristupljeno: 10. 8. 2022.)

[7] Touchscreen gestures - Human Interface Guidelines - Apple Developer, https://developer.apple.com/design/human-interface-guidelines/inputs/touchscreen-gestures (pristupljeno: 7. 7. 2022.)

[8] Gestures - Material Design, https://material.io/design/interaction/gestures.html (pristupljeno: 7. 7. 2022.)

[9] A. Cooper, R. Reimann, D. Cronin, C. Noessel (2014), About Face: The Essentials of Interaction Design, 4th Edition, Wiley,

[10] Pros and Cons of Building Single Page Applications in 2022 - Gearheart, https://gearheart.io/articles/pros-and-cons-building-single-page-applications-2019/ (pristupljeno: 10. 8. 2022.)

[11] What Is a Web Application? How It Works, Benefits and Examples, https://www.indeed.com/career-advice/career-development/what-is-web-application (pristupljeno: 1. 8. 2022.)

[12] Introduction to progressive web apps - MDN, https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps/Introduction (pristupljeno: 10. 8. 2022.)

[13] News from WWDC22: WebKit Features in Safari 16 Beta - WebKit, https://webkit.org/blog/12824/news-from-wwdc-webkit-features-in-safari-16-beta/#web-push-for-macos (pristupljeno: 12. 8. 2022.)

[14] Usage Statistics of Client-side Programming Languages for Websites, August 2022, https://w3techs.com/technologies/overview/client_side_language (pristupljeno: 4. 8. 2022.)

[15] Stack Overflow Developer Survey 2022, https://survey.stackoverflow.co/2022/#most-popular-technologies-webframe (pristupljeno: 4. 8. 2022.)

[16] The State of JS 2021: Front-end Frameworks, https://2021.stateofjs.com/en-US/libraries/front-end-frameworks/ (pristupljeno: 4. 8. 2022.)

[17] Usage Statistics and Market Share of Server-side Programming Languages for Websites, August 2022, https://w3techs.com/technologies/overview/programming_language (pristupljeno: 4. 8. 2022.)

[18] Usage Statistics and Market Share of Content Management Systems, August 2022, https://w3techs.com/technologies/overview/content_management (pristupljeno: 4. 8. 2022.)

[19] The State of JS 2021: Back-end Frameworks, https://2021.stateofjs.com/en-US/libraries/back-end-frameworks/ (pristupljeno: 4. 8. 2022.)

[20] Out-of-Tree Platforms · React Native, https://reactnative.dev/docs/out-of-tree-platforms (pristupljeno: 3. 8. 2022.)

[21] Supported platforms - Flutter, https://docs.flutter.dev/development/tools/sdk/release-notes/supported-platforms (pristupljeno: 3. 8. 2022.)

[22] Professional MultiTouch Tables for Businesses, https://www.mmt.io/touchscreen-tables/ (pristupljeno: 29.8.2022.)

[23] Interactive Experience Table, https://www.logando.de/wp-content/uploads/2020/02/MMT_IET_PCAP_55_65_TABLE_DataSheet.pdf (pristupljeno: 29.8.2022.)

[24] B. Frost (2016), Atomic Design, Brad Frost, , Pittsburgh, Pennsylvania

[25] A. Boduch, R. Derks, M. Sakhniuk (2022), React and React Native: Build cross-platform JavaScript applications with native power for the web, desktop, and mobile, , 4th Edition, Packt Publishing, , Birmingham, UK

[26] Git, https://git-scm.com/ (pristupljeno: 29.8.2022.)

[27] Node.js, https://nodejs.org/ (pristupljeno: 29.8.2022.)

[28] npm, https://www.npmjs.com/ (pristupljeno: 29.8.2022.)

[29] Visual Studio Code - Code editing. Redefined., https://code.visualstudio.com/ (pristupljeno: 29.8.2022.)

[30] Trello, https://trello.com/ (pristupljeno: 29.8.2022.)

[31] @use-gesture, https://use-gesture.netlify.app/ (pristupljeno: 30.8.2022.)

[32] react-spring, https://react-spring.dev/ (pristupljeno:)

[33] styled-components, https://styled-components.com/ (pristupljeno: 30.8.2022.)

6. PRILOZI

6.1. Popis slika

6.2. Popis isječaka koda