Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 4.41
DOI: https://doi.org/10.24867/33BE38Rajnovic

MIGRACIJA MIKRO KLIJENTSKIH APLIKACIJA IZ VEB U DESKTOP OKRUŽENJE

MIGRATION OF MICRO-FRONTEND APPLICATIONS FROM WEB TO DESKTOP ENVIRONMENT

Teodora Rajnović, *Fakultet tehničkih nauka, Novi Sad*

Oblast – SOFTVERSKO INŽENJERSTVO I INFORMACIONE TEHNOLOGIJE

Kratak sadržaj – *U radu je predstavljena arhitektura mikro klijentskih aplikacija (eng. micro-frontend) i prednosti koje ovaj pristup donosi u razvoju veb aplikacija. Opisani su savremeni pristupi u razvoju desktop aplikacija, pri čemu je Electron identifikovan kao najprikladnije rešenje za migraciju postojeće mikro klijentske aplikacije u desktop okruženje. Prikazana je softverska arhitektura konkretne veb aplikacije i proces njene transformacije u desktop aplikaciju, uz očuvanje modularnosti, internet konekcije i postojećeg koda.*

Ključne reči: *Mikro klijentska arhitektura, Desktop aplikacije, Electron radni okvir*

Abstract – *The paper presents the architecture of micro-frontend applications and the advantages that this approach brings in the development of web applications. Modern approaches in desktop application development are described, with Electron identified as the most suitable solution for migrating an existing micro-frontend application to a desktop environment. The software architecture of a specific web application and the process of its transformation into a desktop application are presented, while preserving modularity, internet connection and existing code.*

Keywords: *Micro-frontend architecture, Desktop applications, Electron framework*

1. UVOD

Micro-frontend arhitektura postaje sve popularniji pristup u razvoju modernih i savremenih veb aplikacija. Ovaj pristup omogućava podelu velikih monolitnih klijentskih aplikacija na manje, nezavisne module, što doprinosi lakšem održavanju, unapređenju i timskom radu. Micro-frontend koncept koristi principe mikro-servisne arhitekture, ali se primenjuje na frontend deo aplikacije [1]. Iako su veb aplikacije danas dominantne zbog svoje dostupnosti preko veb pregledača, korisnici informacionih sistema često preferiraju desktop aplikacije zbog boljih performansi, integracije sa operativnim sistemom i veće kontrole nad bezbednosti i privatnosti podataka.

U slučajevima kada su veb aplikacije realizovane kroz upotrebu micro-frontend tehnologije, mogu se uočiti značajni benefiti u kontekstu velikih sistema koji obrađuju velike količine informacija i vrše komunikaciju između različitih komponenti. Ipak, zbog specifičnih zahteva korisnika, kao što su brzina, rad van mreže i direktan pristup hardveru, javlja se potreba za migracijom u desktop okruženje.

Migracija micro-frontend veb aplikacije prikazana je na primeru DevAdmin aplikacije, čija je namena konfiguracija, dijagnostika i analiza hardverskih uređaja. DevAdmin je razvijena kao Angular aplikacija sa modularnom arhitekturom, koja omogućava efikasno upravljanje komponentama i funkcionalnostima. Korisnicima je omogućeno da podešavaju mrežne i bezbednosne parametre, upravljaju licencama i pristupaju statusnim izveštajima radi rešavanja tehničkih problema.

Ovaj rad istražuje tehnološke aspekte te migracije, uz očuvanje modularne arhitekture i postojećeg koda, kao i prednosti koje se postižu u pogledu performansi, bezbednosti i korisničkog iskustva.

2. TEHNOLOŠKE OSNOVE

Primer migracije micro-frontend aplikacije u desktop aplikaciju predstavljen je na primeru Angular [2] aplikacije, uz primenu micro-frontend arhitekture zasnovane na Module Federation konceptu. U svrhu odabira najpogodnijeg rešenja za migraciju iz micro-frontend u desktop okruženje, dat je sažet prikaz savremenih pristupa u razvoju desktop aplikacija.

2.1. Angular

Angular [2] je TypeScript [3] radni okvir koji omogućava izgradnju dinamičkih, single-page aplikacija kroz upotrebu komponenti, šablona i reaktivnog programiranja. Komponente definišu izgled i ponašanje aplikacije, dok servisi omogućavaju deljenje logike i podataka. Pruža podršku za reaktivno programiranje putem signala, odloženo učitavanje komponenti radi poboljšanja performansi, kao i sistem za rutiranja za navigaciju bez ponovnog učitavanja stranice. Angular CLI (Command-line interface) olakšava razvoj aplikacija kroz brzo generisanje elemenata, a njegova arhitektura obezbeđuje strukturiran i održiv razvoj modernih aplikacija.

2.2. JavaScript

JavaScript je interpretirani programski jezik visokog nivoa, uveden 1995. godine, koji omogućava manipulaciju korisničkog interfejsa u veb pregledačima [4]. JavaScript podržava više programskih paradigmi, uključujući i objektno i funkcionalno programiranje, i poseduje ugrađene API (Application Programing Interface) interfejse, za rad sa tekstom, datumima i DOM (Document Object Model) objektima. Iako direktno ne podržava ulazno/izlazne operacije, okruženja poput Node.js proširuju njegovu primenu van pregledača. ECMAScript

standard je nastao kao posledica potrebe za proširenjem, unapređenjem konzistentnosti i standardizacijom JavaScript programskog jezika u različitim okruženjima.

2.3. Micro-frontend aplikacije

Micro-frontend arhitektura predstavlja savremeni pristup razvoju veb aplikacija, koji omogućava podelu klijentske aplikacije na manje, nezavisne i lako upravljive module [5]. Za razliku od tradicionalne arhitekture klijentskih aplikacija, gde je aplikacija monolitnog karaktera i zahteva visok njivo koordinacije među članovima tima, micro-frontend omogućava timovima da samostalno razvijaju, testiraju i primenjuju funkcionalnosti, što značajno unapređuje skalabilnost i brzinu isporuke novih funkcionalnosti. Ovaj pristup je inspirisan mikro-servisima na serverskoj strani i omogućava veću fleksibilnost, nezavisnu primenu i lakše održavanje aplikacija. Velike kompanije poput Spotify, Ikea i Zalando već uspešno primenjuju micro-frontend arhitekturu za razvoj kompleksnih sistema [6].

Za implementaciju micro-frontend aplikacija koriste se alati poput Webpack, Module Federation, Single-SPA i Angular Elements. Module Federation omogućava učitavanje nezavisnih modula, kao što bi na primeru veb aplikacija za kupovinu to bile korisničke korpe za kupovinu ili sistem za plaćanje. Ovakvom arhitekturom izbegava se dupliranje koda i omogućava se deljenje UI (User Interface) komponenti između više aplikacija. Ukoliko se ispoštuju konvencije imenovanja i izoluje kod radi izbegavanja konflikata, ova arhitektura podstiče timsku autonomiju, jer svaki tim može da radi na svom modulu bez potrebe za koordinacijom sa drugim timovima.

Micro-frontend arhitektura se može organizovati kroz dve podele: horizontalna i vertikalna. Horizontalna podela podrazumeva više modula na istoj stranici, gde se timovi fokusiraju na tehničke aspekte poput UI komponenti ili API integracije. Ovaj model je pogodan za velike timove i poslovne poddomene, ali zahteva veću koordinaciju. Vertikalna podela organizuje aplikaciju oko poslovnih funkcionalnosti, gde svaki micro-frontend predstavlja jednu celinu koju razvija jedan tim. Shell aplikacija upravlja učitavanjem ovih modula i obezbeđuje da se, u datom trenutku, prikazuje samo jedan micro-frontend, što je posebno korisno kod SPA aplikacija.

Kompozicija micro-frontend aplikacija omogućava integraciju nezavisnih modula u jedan korisnički interfejs, uz podršku radnih okvira kao što su Angular, React i Vue.js. Web Components omogućavaju kreiranje inkapsuliranih HTML (Hyper Text Markup Language) elemenata, čime se postiže modularnost i lakša saradnja među timovima.

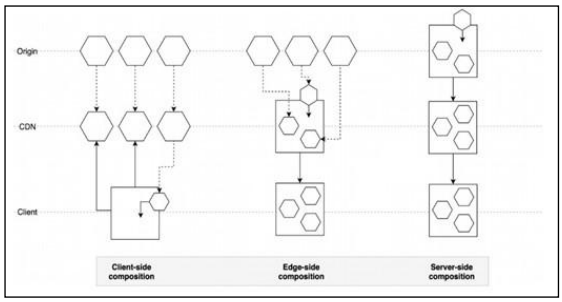
Postoje tri strategije kompozicije micro-frontend aplikacija:

• kompozicija na strani servera – server sastavlja micro-frontend module pre nego što pošalje konačni HTML klijentu, čime se ubrzava učitavanje i smanjuje opterećenje na pregledaču. Idealna je za visoko indeksirane ili kritične veb stranice, ali zahteva tesnu integraciju sa serverom, što ograničava nezavisnost modula.

• kompozicija na strani ivice (edge) – slična je serverskoj kompoziciji, ali se izvršava bliže korisniku, konkretno preko CDN (Content Delivery Network) infrastrukture. Omogućava brzu isporuku sadržaja, ali unosi dodatnu složenost zbog različitih CDN implementacija i ograničenog broja alata za podršku.

• kompozicija na strani klijenta – moduli se učitavaju i sklapaju direktno pomoću shell aplikacije. Ovaj pristup omogućava dinamičko upravljanje, veću autonomiju timova i bolje korisničko iskustvo, bez potrebe za ponovnim učitavanjem cele stranice.

Strategije kompozicije micro-frontend arhitekture prikazane su na slici 1.



Slika 1. *Metode kombinacije micro-frontend arhitektura [6]*

Module Federation je tehnologija koja je uvedena u Webpack verziji 5 i koja omogućava deljenje koda i zavisnosti između više aplikacija [7]. Aplikacije se dele na nezavisno razvijene, testirane i primenjene micro-frontend module (remotes), koji mogu biti ili "proizvođači" (remotes) i "potrošači" (hosts) ili imati obije uloge. Proizvođači izlažu svoje module, a potrošači su aplikacije koje koriste te module. Ovaj pristup smanjuje dupliranje koda, poboljšava performanse i omogućava skalabilan razvoj nezavisnih micro-frontend modula.

2.4. Savremeni pristup u razvoju desktop aplikacija

Desktop aplikacije se izvršavaju lokalno na korisnikovom uređaju, bez potrebe za internet konekcijom, što im omogućava bolje performanse, veću kontrolu nad resursima i veću bezbednost. Iako imaju prednosti u stabilnosti i brzini, zahtevaju instalaciju, redovno ažuriranje i često su vezane za određeni operativni sistem [8].

Razvoj desktop aplikacija uključuje različite tehnologije, zavisno od potreba samog projekta. Pri odabiru tehnologije za realizaciju migracije micro-frontend aplikacije u desktop aplikaciju razmatrani su aspekti poput stepena izmene koda, podrške za složenim funkcionalnostima i multiplatformske podrške. U svrhu migracije i prema datim kriterijumima razmatrane su sledeće tehnologije:

• WPF (.NET) – nudi bogat skup UI elemenata, ali je vezan za Windows. Zahteva potpun prelaz na C# i visok stepen refaktorizacije.

• JavaFX – omogućava razvoj u Java programskom jeziku; slaba integracija sa modernim veb stekom. Zahteva potpunu refaktorizaciju.

• PyQT – dobar GUI alat u Python programskom jeziku, ali zahteva napuštanje postojećeg koda i ima ograničenja u sistemskoj integraciji.

• Flutter – namenjen prvenstveno za mobilne aplikacije, koristi programski jezik Dart. Veliki stepen

refaktorizacije i nije pogodan za kompleksne desktop sisteme.

• Tauri – moderan i lagan okvir sa dobrim performansama, ali manje zreo za složene aplikacije i zahteva prelaz na Rust programski jezik.

• PWA (Progressive Web Apps) – nizak stepen refaktorizacije, ali ne zadovoljava uslove za offline rad sa hardverom.

• Electron – srednji stepen refaktorizacije, omogućava ponovnu upotrebu Angular koda. Podržava multiplatformski razvoj, bogate sistemske funkcionalnosti, ali ima veće memorijsko opterećenje.

Za migraciju postojeće veb aplikacije u desktop okruženje, izabran je Electron radni okvir. Electron omogućava ponovnu upotrebu postojećeg Angular koda, podržava više operativnih sistema i nudi stabilnost i funkcionalnost potrebnu za kompleksne aplikacije. Uprkos većem memorijskom opterećenju, njegova zrelost i široka primena čine ga pouzdanim izborom.

2.5. Electron

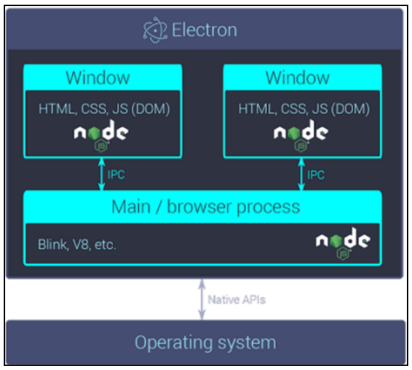
Electron je multiplatformski radni okvir koji omogućava razvoj desktop aplikacija koristeći veb tehnologije kao što su HTML, CSS i JavaScript [9]. Izgrađen je na Chromium i Node.js projektima, što mu omogućava pristup sistemskim resursima i funkcionalnostima kao što su dijalozi, obaveštenja i automatska ažuriranja. Podržava Windows, macOS i Linux, čime obezbeđuje konzistentno korisničko iskustvo na više platformi.

Electron aplikacije su sačinjene od dva procesa:

• glavni proces - upravlja logikom aplikacije, prozorima, sistemskim funkcijama i životnim ciklusom aplikacije.

• proces prikazivanja – upravlja veb sadržajem svakog prozora.

Komunikacija između ovih proces se vrši preko IPC (Inter-Process Communication) mehanizma, što omogućava stabilnost i izolaciju procesa. Glavni proces koristi module kao što su BrowserWindow modul za kreiranje prozora i app modul za upravljanje životnim ciklusom aplikacije. Procesi prikazivanja koriste HTML, CSS i JavaScript za prikaz sadržaja, ali, iz bezbednosnih razloga, nemaju direktan pristupa Node.js API interfejsima. Slika 2 prikazuje arhitekturu Electron aplikacije, kao i odnos između glavnog i procesa prikazivanja.

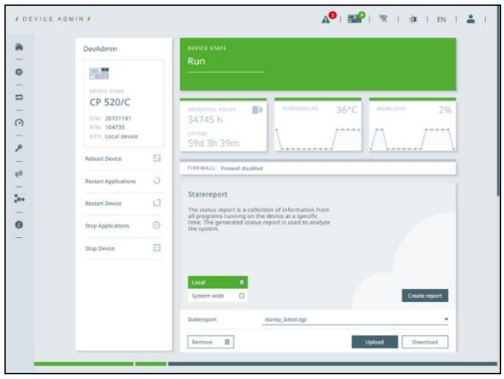


Slika 2. *Arhitektura Electron aplikacije [10]*

Da bi se omogućila bezbedna interakcija između proces, Electron koristi preload skripte koje se izvršavaju pre učitavanja veb sadržaja. Preload skripte imaju pristup Node.js API interfejsima i preko contextBridge modula mogu bezbedno izložiti funkcionalnosti procesu prikazivanja. Pored toga, Electron podržava kreiranje uslužnih procesa preko UtilityProcess API interfejsa, koji se koriste za izvršavanje zahtevnih ili nepouzdanih zadataka. Procesi prikazivanja komuniciraju sa uslužnim procesima preko MessagePort kanala.

3. MIGRACIJA ANGULAR VEB APLIKACIJE U STANDALONE ELECTRON APLIKACIJU

Migracija Angular veb aplikacije u standalone Electron okruženje prikazana je kroz primer DevAdmin aplikacije. DevAdmin je alat za korisničku podršku i analizu hardverskih uređaja. Aplikacija omogućava pregled sistemskih informacija, upravljanje podešavanjima, generisanje izveštaja o statusu i greškama, kao i direktnu interakciju sa uređajima preko lokalne mreže. Zahvaljujući micro-frontend arhitekturi i intuitivnom interfejsu (slika 3.), DevAdmin pruža fleksibilno i efikasno rešenje za tehnički podršku, a njena migracija u Electron aplikaciju omogućava stabilniji i nezavisniji rad u desktop okruženju.

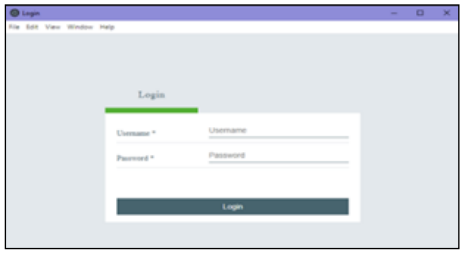


Slika 3. *Pregled interfejsa DevAdmin aplikacije*

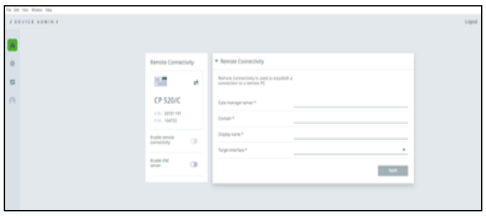
DevAdmin aplikacija predstavlja konkretan primer primene micro-frontend arhitekture uz korišćenje Module Federation koncepta u okviru Angualr radnog okvira. Aplikacija je organizovana kao modularna klijentska veb aplikacije, sa dve shell aplikacije koje dinamički učitavaju više nezavisnih remote modula. Ovaj pristup omogućava timovima da razvijaju funkcionalnosti izolovano, uz bolju skalabilnost i održivost koda. Migracija DevAdmin aplikacije koristeći Electron omogućila bi korišćenje postojećeg veb koda za multiplatformsku distribuciju.

Kompozicija micro-frontend modula u DevAdmin aplikaciji realizovana je na strani klijenta, gde shell aplikacija dinamički integriše remote module tokom izvršavanja. Module Federation omogućava da se ovi moduli učitaju na zahtev, čime se postiže fleksibilna i prilagodljiva arhitektura. Da bi se konfigurisao Module Federation potrebno je definisati webpack.config.js, module-federation.config.json i default-module-federation-config.js fajlovi, u kojima se definišu remote aplikacije i deljeni Angular moduli. Ovaj sistem omogućava optimizaciju učitavanja i izbegavanje konflikta u zavisnostima, čime se unapređuju performanse i stabilnost aplikacije.

Migrirana DevAdmin Electron aplikacija sačinjena je od prozora za prijavu korisnika (slika 4) i glavnog prozora (slika 5). Shell i remote aplikacije se mogu učitati bilo putem udaljene veb adrese ili direktno sa lokalnog diska. Korišćenjem Module Federation koncepta, omogućena je dinamička integracija nezavisnih modula, dok Electron omogućava njihovo prikazivanje u odvojenim karticama ili prozorima.



Slika 4. *Stranica za prijavu korisnika*



Slika 5. *Glavni ekran sa učitanom remote aplikacijom*

Učitavanje aplikacija realizovano je kroz dve metode: preko loadURL metode za učitavanje sa lokalnog servera ili preko loadFile metode za učitavanje kompajliranih HTML fajlova. U drugom slučaju, micro-frontend aplikacije se učitavaju unutar jednog dashboard.html fajla putem <iframe> elementa, čime se simulira ponašanje shell aplikacije. Ovaj pristup omogućava bolju izolovanost komponenti, poboljšava performanse i bezbednost, ali zahteva ručnu izgradnju aplikacije nakon svake izmene.

Da bi se izbegli problemi sa apsolutnim putanjama unutar <iframe> aplikacija, implementirano je preusmeravanje HTTP i WebSocket zahteva na nivou Electron sesije, koristeći webRequests.onBeforeRequest metode. Ovo omogućava da aplikacije zadrže originalnu logiku komunikacije sa serverom, bez izmene koda. Pored toga, upravljanje životnim ciklusom prozora i korisničkom sesijom realizovano je kroz IPC mehanizam i localStorage, čime se obezbeđuje nezavisnost i izolovanost svake remote aplikacije.

Celokupno rešenje omogućava da se postojeća veb arhitektura lako migrira u desktop okruženje, uz minimalne izmene u kodu i zadržavanje osnovnih principa micro-frontend arhitekture. Ovakva kombinacija tehnologija pruža fleksibilnost, stabilnost i mogućnost daljeg proširenja aplikacije u različitim razvojnim i produkcionim scenarijima.

4. ZAKLJUČAK

U ovom radu je uspešno istražena i demonstrirana migracija Angular micro-frontend veb aplikacija u desktop okruženje uz korišćenje Electron radnog okvira. Zbog svoje zrelosti, podrške za naprednim funkcionalnostima i mogućnosti duboke integracije sa operativnim sistemom, Electron je odabran kao najprikladnije rešenje za migraciju, dok su alternative poput Tauri i PWA odbačene zbog nezrelosti tehnologije, podrške za offline rad i slabe integracije sa operativnim sistemom. Migracija je realizovana bez drastičnih izmena u postojećem kodu, uz očuvanje modularnosti i principa koje propisuje micro-frontend arhitektura.

Tehničke strategije uključuju učitavanje remote aplikacija, upravljanje zavisnostima, preusmeravanje HTTP i WebSocket zahteva i komunikacije preko IPC mehanizma. Electron aplikacija preuzima ulogu nove shell aplikacije, omogućujući postepenu i kontrolisanu migraciju. Iako ovaj pristup donosi određene izazove, kao što su povećana potrošnja resursa i bezbednosni rizici, uz primenu preporučenih praksi, moguće je obezbediti stabilno i bezbedno okruženje.

Rezultati pokazuju da je predloženo rešenje efikasno i održivo za prelazak kompleksne veb aplikacije u desktop okruženje, uz zadržavanje fleksibilnosti, skalabilnosti i agilnog razvoja. Ovakva arhitektura omogućava de se micro-frontend aplikacije pokreću nezavisno, čime se zadržavaju sve prednosti originalnog veb pristupa u kontekstu desktop aplikacija.

5. LITERATURA

[1] Prajwal, Y., Parekh, J. V., & Shettar, R. (2021). A brief review of micro-frontends. United International Journal for Research and Technology, 2(8), 18.

[2] https://angular.dev (pristupljeno u septembru 2025.)

[3] https://www.typescriptlang.org (pristupljeno u septembru 2025.)

[4] Haverbeke, M. (2018). Eloquent javascript: A modern introduction to programming. No Starch Press.

[5] https://martinfowler.com/articles/micro-frontends.html (pristupljeno u septembru 2025.)

[6] Peltonen, S., Mezzalira, L., & Taibi, D. (2021). Motivations, benefits, and issues for adopting micro-frontends: A multivocal literature review. Information and Software Technology, 136, 106571.

[7] https://module-federation.io (pristupljeno u novembru 2025.)

[8] https://www.geeksforgeeks.org/what-is-standalone-application (pristupljeno u novembru 2025.)

[9] https://www.electronjs.org (pristupljeno u novembru 2025.)

[10] Alymkulov, D. (2019). Desktop Application Development Using Electron Framework: Native vs. Cross-Platform. [Bachelor's Thesis]. South-Eastern Finland University of Applied Sciences.

Kratka biografija:

Teodora Rajnović rođena je 1. avgusta 1999. godine u Zvorniku. Osnovnu školu „Stefan Mitrov Ljubiša" završila je u Budvi, 2014. godine. U istom gradu je i završila srednju školu „Danilo Kiš", smer turistički tehničar 2018. godine. Kasnije te godine, upisuje Fakultet tehničkih nauka u Novom Sadu, smer Softversko inženjerstvo i informacione tehnologije. Studije završava u roku, 2022. godine.