Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: (Upisuje redakcija, ostaviti ovaj red)
DOI: (Upisuje redakcija, ostaviti ovaj red)

МИКРОФРОНТЕНД АПЛИКАЦИЈЕ

MICROFRONTEND APPLICATIONS

Никола Мандић, *Факултет техничких наука, Нови Сад*

Област – РАЧУНАРСТВО И АУТОМАТИКА

Кратак садржај – *У овом раду представљен је развој једне веб апликације, за чију имплементацију се користе концепти микрофронтенд архитектуре. Рад прати комплетан развој једне такве апликације, од постављања структуре и конфигурисања овакве архитектуре, пратећих проблема до континуираног достављања и испоруке једне такве апликације.*

Кључне речи: *Веб апликација, React, Микрофронтенд*

Abstract – *This paper presents development of one web application, which implementation is based on concepts of microfrontend architecture. The paper follows complete development of application. At development of application is meant to build a structure and configuration of application architecture, solving issues with configurations. Continious integration and delivery for application are covered in this paper.*

Keywords: *Web application, React, Microfrontend*

1. УВОД

Развојем интернет технологија долази до све већег развоја писања софтвера као веб апликација. Најчешћи модел писања представља клијент сервер модел. Веб апликација представља систем који је направљен користећи клијент-сервер архитектуру [1]. Клијент-сервер модел је значајно смањио развој развој апликације тако што долази до раздвајања функционалности на клијентску и серверску страну [2]. У почетку и клијентска и серверска страна писане су као монолитне апликације. Повећањем пословних захтева система и све тежег одржавња серверске стране модела долази до појаве микросервисне архитектуре. Микросервисна архитектура подразумева рашчлањивање монолитне апликације на више мањих апликација зарад лакшег одржавања и даљег развоја апликација. По узору на микросервисне апликације јавља се идеја да се такав приступ примени и на клијентску страну, која је у почетку формирана као монолитна апликација.

2. МИКРОФРОНТЕНД АПЛИКАЦИЈЕ

Претходних година током развоја веб апликација, започето је коришћење микросервисне архитектуре на серверској страни.

НАПОМЕНА:
Овај рад проистекао је из мастер рада чији ментор је био др Александар Купусинац, ред. проф.

Микросервисна архитектура представља још један стил архитектуре софтверских система у коме се велике сложене апликације компонују склапањем појединачних сервиса [3].

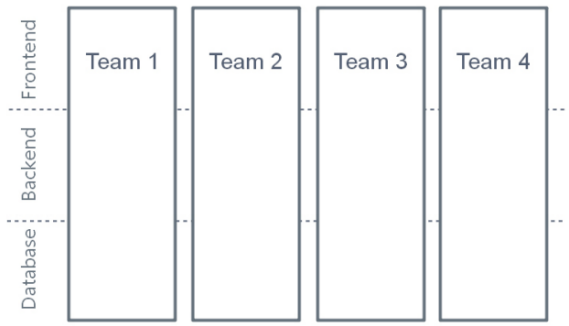
Критеријуми које сервиси задовољавају су [3, 4]:

- Лаки за тестирање и одржавање.
- Слабо спрегнути.
- Сваки сервис обавља један задатак, неки део пословне логике система.
- Могу бити независно *deployovani* .
- Сваки микросервис је могуће развијати у програмском језику који је погодан захтевима.
- Не морају бити свесни имплементације осталих сервиса.
- Комуникација се обавља програмским интерфејсима *API-ijima* независним од програмског језика (нпр. *Representational State Transfer (REST)*)

Микрофронтенди наслеђују концепте микросервиса на страни веб претраживача. Под овим се подразумева да се апликација на клијентској страни раздваја на више мањих апликација [5]. Свака од тих апликација може да се покреће, развија и има независан *deployment* [5]. Бенефите које је донела микросервисна архитектура, попут модуларности, оптимизације, лакшег раста и развоја апликације доводе до развоја идеје да се исти такав коцепт примени и на клијентској страни. Идеја на коју се ослањају микрофронтенди је та да се веб апликација третира као комбинација функционалности везаних за различит тим, то подразумева да сваки тим има задужење да развија одређен сет повезаних пословних функционалности апликације [5]. Тимови су ункарсно функционални и развијају целокупну функционалност, а под тим се подразумева да се развија клијентски и серверски део апликације.

Концепти и идеје на којима се базирају микрофронтенди [1, 5]:

- Сваки тим има слободан избор технологије коју користи.
- Код сваког тима мора бити изолован, подразмева да се не користе глобалне варијабле.
- Коришћење префикса код варијабли да се зна власништво варијабле да се изебгну конфликти.
- Коришћење емитовања догађаја веб претраживача зарад комуникације између апликација



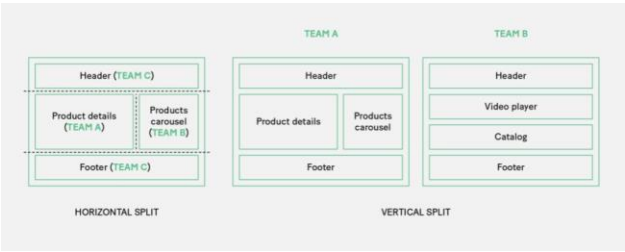
Слика 1. *Организација тимова са микрофронтенд архитектуром* [5]

На слици 1. представљен је начин поделе тимова унутар организације као последица коришћења микрофронтенд архитектуре, сваки тим је задужен за одређен домен система. Под тим се подразумева да тим развија и клијентску и серверску страну у складу са доменом који им је додељен.

Важно је нагласити да овакав приступ у архитектури клијентске апликације отвара потенцијалне проблеме који требају да се реше. Неки од њих су [1]:

- Оркестрација – како решити учитавање апликације само када је потребна.
- Рутирање – како поставити логику рутирања и одлучити коју апликацију учитати.
- Изолација – како избећи колизије у окружењу.
- Комуникација – на који начин апликације треба да комуницирају једна између друге.
- Конзизстентан кориснички интерфејс – обезбедити да једна апликација не утиче на стилове друге апликације.
- Менаџмент билблиотека – како избећи да се једна библиотека учитава више пута.

Микрофронтенд апликације се могу организовати на два начина хоризонтално и вертикално (слика 2). Под хоризонталном организацијом подразумева се да може више апликација бити присутно на једној страници унутар веб претраживача, док под вертикалном организацијом једна апликација се извршава по страници [6]. Једна од напомена приликом коришћења хоризонталног приступа је нарушавање прицнипа да би свака апликација требала имати задужење на основу одређене пословне логике. Вертикална организација је веома слична начину на који функционишу традиционалне *SPA (Single Page Application)*, при чему сваки тим води рачуна о специфичним деловима апликације [6].

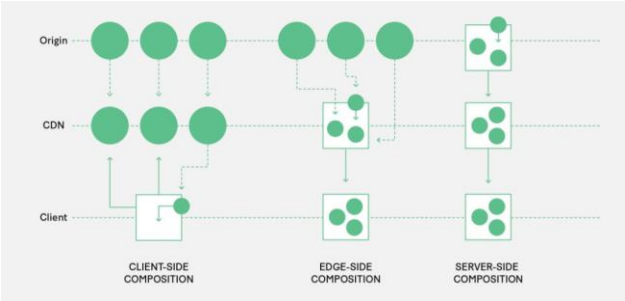


Слика 2. *Хоризонтална и вертикална организација микрофронтенда* [6]

Постоје различити приступи имплементације решења микрофронтенд апликације (слика 3) [1]:

- Композиција на серверској страни – приликом генерисања страница на серверу, врши се оркестрација неопходних апликација. Проблем овог приступа је тај што се губе могућности *SPA (Single Page Application),* јер се при сваком захтеву са клијентске стране морају генерисати странице.
- *Build-time* интеграција – подразумева да се свака апликација генерише као независан *JavaScript* пакет и интегрише се током завршне компилације модула. Главна мана овог приступа је та што сви микрофронтенди морају бити рекомпајлирани ако се било који од њих измени.
- *Run-time* интеграција помоћу *iframe-a* - унутар *iframe* тага се добија изолација апликација. Мана овог приступа је немогућност дељења истих библиотека између апликација, па као последица тога апликација заузима више простора унутар веб претраживача. Такође једини начин комуникације представља *API* од *iframe* што аутоматски смањује флексибилност на пољу комуникације између апликација.
- *Run-time* интеграција помоћу *JavaScript-a* – представља једно од најоптималнијих решења. При овом решењу се подразумева да се свака апликација прави као одвојен модул и да се учитава једино уколико је потребно. Сваки модул се развија независно што омогућава тимовима да равијају функционалности независно од осталих. Са овим приступом омогућено је коришћење заједничких библиотека између апликација.
- *Run-time* интеграција помоћу веб компоненти - представља најновији приступ који постаје могућ због нових *HTML* стандарда. По овим стандардима дозвољава се креирање свог *HTML* елемента, дефинисање понашања тог елемента и динамичног учитавања кода тог елемента. Елемент је по питању изолације на истом нивоу као и *iframe* таг. Предност овог приступа је у комуникацији између апликација што су у овом случају веб компоненте. Проблем код овог приступа је његова старост, јер су стандарди на ком се базира нови и велики број веб претраживача није прилагођен раду по тим стандардима. Постоји начин да се и овај проблем заобиђе, коришћењем *polyfill-a,* који представљају библиотеке које трансформишу у прилагодљив код старијим веб претраживачима. Додатан проблем је што се самим тим повећава количина садржаја сервирана веб претраживачу.

Постоји још и приступ дистрибуције апликација на веб серверу, где се различита апликација учитава у зависности од руте којој се приступа. Мана овог приступа је да се сваки пут страница мора освежити када се приступа новој рути на веб претраживачу.



Слика 3. *Различити приступи имплементације микрофронтенд апликација* [6]

3. КЛАУД КОМПЈУТИНГ

Постоји доста дефиниција за клауд компјутинг, једна од најчешће коришћених је та да клауд компјутинг представља групу дистрибуираних компјутера који пружају сервисе и ресурсе кроз интернет [2].

Постоје три врсте сервиса које нуди клауд компјутинг:

- *IaaS (Infrastructure as a Service),* кориснику је пружена могућност коришћења рачунарске инфраструктуре у виду виртуелне платформе.
- *PaaS (Platform as a Service),* са овом опцијом корисник не треба да води више рачуна попут хардвера, оперативног система. Омогућава кориснику потпуни развој апликације.
- *SaaS (Software as a Service),* под овом тачком пружа се опција коришћења готових софтверских решења који су одржавани од стране пружаоца услуге.

Развој клауд компјутинга допринео је у многоме развоју скалабилности серверских страна веб апликација, убрзава се процес развоја софтвера избацивањем неких раније неопходних ствари попут сервера на физичким локацијама и осталог. Предности клауд компјутинга преставља лаган начин избора технологија и услуга у зависности од проблема који се решава. Поред наведеног, врло лако у зависности од оптерећења система се може скалирати и велика уштеда се јавља по погледу трошкова (нису потребни више физички присутни центри података и др.).

4. ИМПЛЕМЕНТАЦИЈА

Систем представља апликација за преглед и додавање знаменитости. Корисници имају могућност да прегледају споменике на мапи и да позиционирају мапу у односу на споменик, такође имају могућност додавања нових споменика у систем.

Архитектура овог система се базира на клијент сервер моделу. При чему клијентска страна је имплементирана по концептима микрофронтед архитектуре што представља тему овог рада. Серверска страна система имплементирана је у *Node.js* платформи са коришћењем *Express* радног оквира погодног за развијање бекенд дела веб апликација. База коришћења за чување података је *MongoDB,* која спада у ред *NoSQL* база података.

Клијентски део састоји се од три апликације које чине заједно микрофронтенд архитектуру. Једна од апликација задужена је за приказ споменика на мапи, друга апликација служи за функционалности додавања споменика и преглед листе свих споменика. Намена треће апликације служи зарад интегрисања горе наведених апликација и њиховог функционисања и пружања одговарајуће логике рутирања и приказа интегрисаних апликација на основу руте. Трећа апликација се назива и контејнер апликација.

```
const devConfig = {
  mode: 'development',
  output: {
    publicPath: 'http://localhost:8082/',
  },
  devServer: {
    port: 8082,
    historyApiFallback: {
      index: '/index.html',
    },
  },
  plugins: [
    new ModuleFederationPlugin({
      name: 'monuments',
      filename: 'remoteEntry.js',
      exposes: {
        './MonumentsApp': './src/bootstrap',
      },
      shared: packageJson.dependencies,
    }),
    new HtmlWebpackPlugin({
      template: './public/index.html',
    }),
  ],
};

module.exports = merge(commonConfig, devConfig);
```

Слика 4. *Конфигурација Webpack-a*

Конфигурација *Webpack-a* (слика 4) користи се унутар апликација зарад њихове комуникације са осталим апликацијама и формирања извршивог кода разумљивог широком спектру веб претраживача. На основу те конфигурације свака апликација има могућност да се покреће у изолацији са чим је постигнут ефекат да сваки тим може независно у односу на цео систем развијати део пословне логике везане за свој домен. Поред тога све апликације су подешене да раде заједно једна са другом. Поље *name* и *filename* представљају локацију и идентификатор на основу ког контејнер апликација долази до неопходних доступних модула *remote* апликације. *ModuleFederationPlugin* представља функционалност *Webpack-a* која омогућава да се формира микрофронтенд апликација и постизање комуникације између различитих апликација и њиховог извршавања. Унутар конфигурације приказане на слици 4. поље *exposes* служи да се се прикажу делови апликације који су доступни контејнер апликацији за коришћење. То значи када се унутар контејнера захтева приступ апликацији споменика тај захтев ће се мапирати на *bootstrap* датотеку унутар апликације за споменике.

Приступ који се користи за имплементацију клијентске стране представља интеграција у *Run-time* помоћу *JavaScript-a.* Приликом имплементације ових апликација коришћена је *React* библиотека за исцртавање корисничког интерфејса.

Током развоја апликације посвећена је пажња на аспект континуиране интеграције и честог достављања апликације корниснику. За подешавање *CI/CD pipeline* коришћене су *Github Actions.* Конфигурацијом ових акција постиже се резултат да на основу сваког догађаја приликом промене кода унутар апликација окида се акција у којој се извршавају одређени кораци. Неки од корака су креирање *build* верзије апликације и синхронизација нових датотека са неким од клауд провајдера. За клауд провајдера коришћени су Амазонови веб сервиси и широк спектар њихових функционалности за веб апликације. Ова тачка представља један од услова формирања архитектуре где промене унутар једне од апликације које чине клијента ће бити увек освежене и неће утицати на рад осталих апликација.

5. ЗАКЉУЧАК

Овај рад представља креирање, упознавање са процесима и концептима микрофронтенд апликација на клијентској страни веб апликација. Објашњени су приступи начина имплементације оваквих апликација. Решење је имплементирано у виду *JavaScript Run-time* приступа. Предности приступа имплементираног унутар решења су:

- Изолован развој апликација које чине систем.
- Промене настале приликом развоја једне од апликација не захтевају рекомпајлирање свих осталих апликација.
- Све апликације се интегришу у реалном времену.
- Могућност коришћења различитих технологија при имплементацији апликације, интеграција је одрађена на генеричан начин.

Приликом имплементације приказане су неопходне технологије за реализацију апликације. У моменту писања рада, микрофронтенд апликације спадају у новије приступе креирања веб апликација и по мишљењу аутора побољшање апликације и генерално целог приступа представља проналазак што генеричнијег начина комуникације између апликација и напредније конфигурисање *Webpack-a* за постизање што оптимизованијих апликација.

6. ЛИТЕРАТУРА

[1] Pavlenko, Andrey, et al. "Micro-frontends: application of microservices to web front-ends." J. Internet Serv. Inf. Secur. 10.2 (2020): 49-66.

[2] Oluwatosin, Haroon Shakirat. "Client-server model." IOSR Journal of Computer Engineering 16.1 (2014): 67-71.

[3] Мирослав Зарић, *Микросервиси*, https://enastava.ftninformatika.com/courses/257/files/folder/predavanja?preview=112844, Факултет техничких наука, Нови Сад, (Приступљено: 09.2022.)

[4] What are microservices ? Доступно: https://microservices.io/

[5] Yang, Caifang, Chuanchang Liu, and Zhiyuan Su. "Research and application of micro frontends." IOP conference series: materials science and engineering. Vol. 490. No. 6. IOP Publishing, 2019.

[6] Micro-frontends in context, Luca Mezzalira, Доступно: https://increment.com/frontend/micro-frontends-in-context/

Кратка биографија:

Никола Мандић рођен је у Вуковару 1996. год. Основне академске студије завршио је 2019. године на Факултету техничких наука у Новом Саду. Мастер рад на Факултету техничких наука из области Рачунарство и аутоматика – Електронско пословање одбранио је 2022. године.