|  |  |  |
| --- | --- | --- |
|  | УНИВЕРЗИТЕТ У НОВОМ САДУ  **ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА У НОВОМ САДУ** |  |

Марија Кастратовић

GoBooking – портал за туристичку агенцију базиран на микросервисној архитектури

ДИПЛОМСКИ РАД

- Основне академске студије -

Нови Сад, 2022

Образац **Q2.НА.15**-**04** - Издање 1

### Садржај

references, table of contents

### 1 Увод

Тема овог рада је креирање портала за туристичку агенцију – Go Booking који је базиран на микросервисној архитектури. Апликацији могу да приступе регистровани корисници који желе да прегледају доступна путовања и резервишу карту за жељено путовање. Такође, корисницима је омогућено оцењивање дестинација као и отказивање резервација. Приступ апликацији имају и администратори система који имају могућност ажурирања података о дестинацијама и путовањима, као и додавање других админа система.

## 2 ТЕОРИЈСКЕ ОСНОВЕ

Израда решења захтева предзнање о микросервисима, па ће у овом поглављу бити .........

## 2.1 Микросервисна архитектура

Апликација базирана на микросервисној архитектури је апликација која је прављена као скуп лабаво везаних сервиса који имплементирају функционалности апликације, а сами су независно развијани и независно имплементирани. Сваки од микросервиса покренут је у сопственом процесу и комуницирају преко мреже добро дефинисаним и стандардизованим протоколом, обично HTTP(енг. *Hypertext Transfer Protocol*). Микросервиси (модули који их реализују) имају потпуно ограничен контекст - не морају бити свесни никаквих имплементационих детаља и архитектуре других микросервисих модула.

Иако је монолитна архитектура и данас врло популарна приликом израде њеб апликација, она садржи и поједине недостатке који могу послужити као увод у микросервисне архитектуре. Монолитне архитектуре су најчешће сачињене од једне софтверске компоненте која је представљала спрегу између корисничког интерфејса и базе података. Растом и развојем оваквих система јављају се многи проблеми као што су скалирање и одржавање њеб апликације, што резултира великом комплексности програмског кода те отежаним будућим одржавањем програмског кода и надоградњом апликације. Улога микросервисне архитектуре је да се апликација подели на мање делове, односно сервисе који су задужени за решавање специфичних проблема у апликацији. Микросервиси могу бити независно испоручивани у продукцију, док је код монолитних апликација било неопходно испоручити целу апликацију и приликом најмање измене система.

# 3 Коришћене технологије

У овом поглављу ће бити представљене технологије коришћене приликом имплементације решења.

## 3.1 ГO

Го је програмски језик опште намене, развијен у компанији Google. Настао 2007. годинем а јавно представљем 2009. године и као такав спада у младе програмске језике. Настао је са циљем да се направи нови језик опште намене, при чему се водило рачуна да се укомбиују лак начин читања и писања као и сигурност језика који се компајлира. Слична синтакса, али знатно поједностављена у односу на језик Ц, омогућила је великом броју програмера да се лако упознају са овим језиком.

Го је статички типизирани, компајлирани језик и поседује гарбаге-цоллецтор механизам. Сличан је Ц програмском језику али са додацима меморијске безбедности garbage-collection механизма, структурне типизираности и ЦСП (енг. *Communicating Sequential Processes*) конкурентности. Го није објектно-оријентисан, али преузима неке од концепата објектно оријентисаних језика. Брзо компајлирање као и високе перформансе приликом мрежне комуникације чине Го погодним за развој микросервисних апликација. Ово нам омогућава лакши и бржи развој, као и континуирану испоруку сервиса.

## 3.2 ПАЈТОН И ФЛАСК

Пајтон (енг. *Python*) је интерпретирани програмски језик високог нивоа настао почетком деведесетих година двадесетог века. Његов творац, холандски програмер Гвидо Ван Росум, имао је жељу да направи програмски језик у коме ће акценат бити на читљивом и прегледном коду који ће програмери лако разумети. Подржава процедуралну, објектно-оријентисану и функционалну парадигму програмирања. За програмски језик Пајтон развијен је велики број стандардних модула који омогућавају ефикасан рад у многим областима.

Фласк (енг. *Flask*) микро веб развојни оквир (енг. *Microframework* ) написан у програмском језику Python. Flask подржава екстензије које су написане изван самог Flask-а, а могу додати могућности апликацијама попут рада с веб формама, приступа базама података, логовања и слично. Једна од предности коришћења Flask микровеб развојног оквира је његова скалабилност, односно погодан је за креирање мањих апликација које у будућности имају потенцијал да прерасту у велике и сложене апликације.

## 3.3 PHARO

*Pharo* је језик настао по угледу на Smalltalk језик са циљем његовог унапређења. Представља прави објекто-оријентисани језик јер је све објекат односно инстанца класе. Модел је динамички типизиран, типови се знају у рун-тајму. Методе су јавне и виртуалне, али су сви атрибути заштићени. *Pharo* подржава искључио једноструко наслеђивање. Као што је напоменуто у *Pharu* је све објекат, а тим објектима се управља коришћењем порука. Порука може бити било шта што исказује неку намеру или враћа одређено стање објекта. На поруку објекат одговра тако што позива одгповарајућу методу, а метода описује шта треба урадити када стигне одређена порука. Објекат који прима поруку се назива пријемник (*receiver*)

## 3.4 *POSTGRESQL*

PostgreSQL или једноставније Postgres je врста релационих система за управљање базама података (енг. Relational Database Management System – RDBMS), под open source лиценцом (отворени код). Сматра се једном од најпоузданијих база података. Најчешће користи за веб апликације и веб базе података. PostgreSQL садржи моћан објектно-релациони модел података, богат избор врста података, лаку надоградивост, као и надограђени сет наредби SQL језика (engl. *SQL - Structured Query Language*).

SQL представља структуирани упитни језик за приступ и манипулацију подацима.

* Упитни језик (енг. *Query Language – QL*)
* Језик за манипулацију подацима (енг*. Data Manipulation Language – DML*)
* Језик за дефинисање података (енг. *Data Definition Language – DDL*)

Прва главна предност коришц́ења Postgres-a је то што је *open-source* и може се прилагодити према захтевима девелопера. Захтеви за одржавање и подешавање Postgres базе података су релативно мали у односу на друге системе за управљање базама података. Такође, PostgreSQL је портабилан и преносив са готово свим главним платформама и програмским језицима па је самим тим идеална за апликације намењене вишеструким платформама.

## 3.5 *HTTP* ПРОТОКОЛ

*Http* (енг. HyperText Transfer Protocol) предтсавља мрежни протокол који служи за комуникацију између сервера и клијента. Комуникација је заснована на принципу захтев/одговор, при чему је сваки пар захтев/одговор међусобно независан од осталих. Сервер је тај који константно ослушкује захтеве на одговарајућем комуникацијском порту и чека да се клијент повеће и пошаље свој захтев. Захтев који је клијент послао се обрађује на серверу, при ћему се клијенту шаље одговарајући одговор на захтев. Http захтев, приказан на слици 3.1, у првом реду садржи податке о методи, путањи и верзији протокола. Затим следе додатни редови који садрже атрибуте облика кључ-вредност. Након тога се налази празан ред и тело поруке које је опционо.



Слика 3.1 Пример HTTP захтева

*Http* дефинише скуп метода захтева за означавање жељене радње коју треба извршити за дати ресурс. Постоји укупно осам Http захтева, при чему су најзаступљенији:

* **GET** – Захтева ресурс од веб сервера
* **HEAD** - Захтева само Http одговор, без слања самог ресурса
* **POST** – Шаље податке серверу, при чему се најчешће креира нови ресурс
* **PUT** – Замењује тренутну репрезентацију ресурса новом која се налази у захтевз
* **DELETE** – Брише све тренутне репрезентације датог ресурса
* **OPTIONS** – Од сервера тражи списак метода које подржава

Методе HEAD, OPTIONS и GET означене су као сигурне методе јер не врше измене на серверу, већ само имају могућност прегледа ресурса. С друге стране методе POST, PUT и DELETE се сматрају несигурним методама јер могу довести до промена стања сервера.

## 3.6 АНГУЛАР

Ангулар је платформа и радни oквир који се користи за креирање апликација са једном страном (енг*. SPA - Single Page Application*) . Ангулар развојно окружење се користи за креирање клијентског дела веб апликације (енг. Client-side framework) коришћењем *HTML*-а, *CSS*-а *иTypeScript-a*. Основни градивни блок сваке Ангулар апликације су модули (NgModule).

Модул се састоји од компоненти, а једна Ангулар компонента обезбеђује део функционалности за апликацију и контролише део екрана који се назива поглед.

Компонента се састоји од:

* Класе
* Метаподатака који описују класу и проширују њену функционалност
* Шаблона који се користи за дефинисање *HTML* погледа

Поглед представља низ елемената на страници којима Ангулар може да приступа и да их модификује. Компоненте користе сервисе, који обезбеђују специфичне функционалности. Сервиси се могу уметнути у компоненте као зависности, при чему доприносе модуларности, чине код поновно искористивим и ефикаснијим.

*Data Binding* је процес прослеђивања података из компоненте ка погледу и обрнуто. Врши се повезивање DOM (енг. *Document Object Model*) елемената са својствима класе компоненте . Постоји 4 начина повезивања података у Ангулару:

* Интерполација
* Повезивање својства (енг. *Property binding*)
* Повезивање догађаја (енг. *Event binding*)
* Двосмерно повезивање (енг. *Two Way Binding*)

## Литература

1. Angular -> <https://angular.io/guide/architecture>
2. Http -> <https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods>
3. slika1 <https://www.cobalt.io/hs-fs/hubfs/Imported_Blog_Media/image-http-request-smuggling-1-1.png?width=475&name=image-http-request-smuggling-1-1.png>
4. slika2 <https://www.cobalt.io/hs-fs/hubfs/Imported_Blog_Media/image-http-request-smuggling-3-1.png?width=680&name=image-http-request-smuggling-3-1.png>