

Студијски програм: Информатика

Предмет: Програмски језици

РЕСТОРАН

- Пројекат –

|  |  |
| --- | --- |
| Предметни наставник: | Студент: |
| др Александар Мишковић | Момир Недељковић 036/2022 |

Крагујевац 2024.

# Поставка задатка

Циљ пројекта: Развити RESTful web сервис за управљање активностима у ресторану, укључујући евиденцију јела и пића, резервације столова, наруџбине хране и пића и праћење залиха.

Функционалности:

а) Евиденција јела и пића: Категоризација јела (пређела, главна јела, десерти) и пића (топли напици, безалкохолна и алкохолна пића), са детаљима као што су састојци, цена и препоручене порције.

б) Резервације столова: Могућност резервације столова, укључујући одабир датума, времена и броја особа.

ц) Наруџбине хране и пића: Омогућити гостима да наручују храну и пиће директно преко сервиса (не узимамо у обзир конобаре), уз праћење статуса наруџбине.

д) Праћење залиха: Аутоматско ажурирање залиха на основу наруџбина и потрошње, уз обавештења о ниским залихама (уколико је то мање од 20%).

е) Обавештења: Имплементација додатне нити која периодично проверава и шаље обавештења о резервацијама, наруџбинама и стању залиха (то се записује у .txt датотеку/датотеке).

Технички Захтеви:

а) Backend: Развој backend апликације користећи Spring Boot 3.0, МySQL и ЈDK 17.

б) Логовање: Имплементација логовања за праћење активности корисника и система.

ц) Multi-layer: Имплементација вишеслојне архитектуре.

д) ЈSON: Комуникација између клијента и сервера користећи ЈSON формат.

е) Тестирање: Тестирање сервиса коришћењем Postman-a.

Структура Података:

ф) Јело: ID јела, назив, категорија (пређело, главно јело, десерт), опис јела и цена.

г) Пиће: ID пића, назив, тип (топли напитак, безалкохолно, алкохолно) и цена.

х) Резервација: ID резервације, датум, време, број особа, контакт информације.

и) Наруџбина: ID наруџбине, детаљи јела и пића, укупан износ и статус наруџбине.

ј) Залихе: ID артикла, назив и количина на залихама.

к) Остале класе и интерфејси: Пројекат садржи више класа разврстаних духу вишеслојне архитектуре (Controller, Component, Service, Repository, DAO, Logger, ... у зависности од реализације вашег пројекта).

Документација: Детаљна документација о АPI endpoint-ovima, примери употребе и логови активности. Писати је према приложеном упутству.

Завршна напомена: Пројекат треба да илуструје способност развоја комплексних RESTful web сервиса, ефикасно управљање базама података, примену вишенитног програмирања и аспектног програмирања у контексту Spring Boot апликација.

# Садржај

# Увод

**1.1 Spring Framework**

**Spring Framework** је радни оквир за платформу Java који обезбеђује инфраструктуру за развој апликације. Омогућава лакше и брже креирање софтвера, чистији код на јави, а тиме и лакше одржавање апликације. Spring је организован у око 20 модула које је могуће појединачно употребљавати зависно од потреба апликације и отвореног је кода. Прву верзију је написао Род Џонсон, а представио је крајем октобра 2002. У јуну 2003. оквир је први пут објављен под лиценцом Apache 2.0 license. Прво издање за продукцију (верзија 1.0) објављено је у марту 2004. године. Верзија 1.2.6 освојила је награду за продуктивност Jolt, али и награду за иновацију године JAX. Верзија 4.0 обухвата значајна побољшања, попут подршке за Java SE 8, Groovy 2, неке аспекте Java EE 7 и WebSocket.

* 1. **О пројекту**

У овом семинарском раду ће бити представљен развој RESTful web servis-a за ресторан користећи Spring Boot 3.0 и JDK 17. Сервис ће омогућити управљање активностима у ресторану и наруџбинама. Подаци ће бити смештени у MySQL бази података, а Maven ће се користити за управљање зависностима.

# 2. Теоретска основа

**2.1 RESTfull архитектура**

REST (Representational State Transfer) је архитектурни стил дизајниран за лакшу комуникацију између дистрибуираних система.RESTfull је као web сервис дефинисан 2004. године. Данас га користе многи популарни web портали као што су:Facebook I Twitter. Кључни принципи RESTfull архитектуре укључују:

* Ресурси: представљају ентитете са својствима које се могу манипулисати (нпр.jело, пиће, резервације ...).
* HTTP Методе: коришћење стандардних HTTP метода (GET, POST,DELETE) за манипулацију ресурсима.
* Репрезентација: ресурси се преносе као репрезентације, често у JSON формату.
* Stateless (Безстање): сваки захтев од клијента садржи све информације потребне за обраду захтева, омогућавајући скалабилност и смањење зависности.

**2.2 Spring Boot**

Spring Boot пружа једноставан начин за развој Spring апликација. Кључне карактеристике укључују:

* Аутоматско конфигурисање: Spring Boot аутоматски конфигурише апликацију према конвенцијама, смањујући потребу за ручним подешавањем.
* Уграђен Tomcat сервер: уграђени контејнер који омогућава самостално извршавање апликације.
* Spring Data JPA: интеграција са JPA за олакшано управљање подацима.

**2.3 Вишеслојна архитектура**

Вишеслојна архитектура је приступ организацији кода у различите слојеве како би сепостигла боља модуларност и одрживост:

* Controller sloj: прима HTTP захтеве, обрађује их и позива одговарајуће методе сервиса.
* Service слој: пословна логика, извршава одређене задатке.
* Repository Слој: комуницира с базом података, извршава операције над ентитетима.

**2.5 MySQL база података**

Spring Boot се лако интегрише са MySQL базом података користећи Spring Data JPA. База података ће садржавати табеле за јело, наруџбине, пиће и остало.

# 3. Опис практичног дела

Вишеслојна архитектура представља организацију софтверске апликације у неколико слојева, где сваки слој има специфичну одговорност. Ова пракса има за циљ побољшање модуларности, одржавања и скалабилности система. У контексту развоја RESTfull web сервиса за управљање рестораном, примењена је вишеслојна архитектура како би се постигли следећи кључни аспекти:

1. Контролни слој (Controller):

* Овај слој је одговоран за примање HTTP захтева и управљање током захтева кроз апликацију.
* Контролери дефинисани у овом слоју обављају валидацију, обраду захтева и позивају одговарајуће методе сервисног слоја.
* Примењена анотација **@RestController** у Spring Boot-у означава ове

компоненте.

1. Сервисни слој (Service):

* Сервиси садрже пословну логику апликације.
* Сервиси позивају методе репозиторијума за приступ подацима из базе.

1. Репозиторијски слој (Repository):

* Репозиторијуми представљају интерфејсе или класе које омогућавају приступ подацима у бази.
* Коришћењем Spring Boot JPA, репозиторијуми олакшавају рад са базом података кроз аутоматски генерисане SQL упите.

1. Моделни слој (Eнтитети):

* Моделни слој садржи ентитете који представљају структуру података у апликацији.
* Јава класе попут Јело, пиће користе се за мапирање ка бази података.

Организацијом система у ове јасно дефинисане слојеве, постижу се следеће предности:

* Модуларност: сваки слој има одвојену одговорност, што олакшава додавање, измену или брисање функционалности без утицаја на остатак система.
* Одржавање: раздвајање одговорности појединих слојева чини одржавање и праћење грешака ефикаснијим.
* Тестирање: сваки слој може се тестирати независно, што олакшава имплементацију и извођење тестова.
* Скалабилност: апликација постаје лакше скалабилна јер сваки слој може

бити оптимизован засебно.

JSON комуникација:

У Spring Boot апликацијама, комуникација између клијента и сервера често се одвија у JSON формату. Spring Boot пружа уграђену подршку за аутоматско мапирање Јава објеката у JSON формат (серијализација) приликом слања података клијенту, као и обрнуто (десеријализација) приликом пријема података са клијента. Ово је постигнуто коришћењем Jackson библиотеке, која је популарна библиотека за рад са JSON форматом у Јава апликацијама. Када клијент упути захтев за добијање детаља наруџбенице, Spring Boot аутоматски користи Jackson библиотеку да серијализује објекат Нарудџбеница у JSON формат пре слања као одговор. Када клијент пошаље POST захтев са JSON подацима који представљају нову наруџбеницу, Spring Boot ће користити Jackson библиотеку да десеријализује JSON податке у објекат Нарудџбеница пре него што их проследи сервису за даљу обраду. Овај процес аутоматског мапирања између Јава објеката и JSON формата значајно олакшава рад са подацима у RESTful апликацијама, побољшава читљивост кода и повећава ефикасност развоја.

**Aнотације**

@Id - означава да је поље на које је стављена примарни кључ (ID) у табели базе података.

@GeneratedValue - ово је додатна анотација која омогућава аутоматско генерисање ID вредности приликом уноса новог ентитета у базу података.

@RestController - означава класу која ће служити као RESTfull web сервис.

@Entity - када користимо @Id, она обично иде у склопу са @Entity, што означава да је класа мапирана на табелу у бази података.

@RequestBody - oзначава да се подаци из HTTP захтева (у ЈSON формату) аутоматски конвертују у Јава објекат.

@PathVariable - Означава да променљива у путањи (URL) може бити повезана са параметром методе.

@Table - користи се за специфицирање детаља о табели на коју је мапиран ентитет.

@Column - користи за мапирање поља (атрибута) унутар Јава класе на колоне у табели базе података.

@Override - користи када желимо да препишемо (оverride) методу из надкласе (суперкласе) или да имплементирамо методу из интерфејса

@Service - a klasu kao **servisni sloj** (service layer) u aplikaciji.

@Autowired - користи се за аутоматску ињекцију зависности (dependency injection).

@RequestMapping - користи се за мапирање HTTP захтева на одређене методе или класе које ће обрадити те захтеве.

@GetMapping, @PostMapping, i @DeleteMapping - су скраћенице у Spring Framework-у које се користе за мапирање специфичних HTTP метода (GET, POST, DELETE) на контролерске методе.

@Component - означава класу као bean који ће бити аутоматски препознат и регистрован у Spring-овом контексту

@Scheduled - користи се за заказивање аутоматског извршавања метода у унапред дефинисаним интервалима.

Postman тестирање:

Тестирање је кључна фаза развоја било које апликације, укључујући и RESTfull web сервисе. У овом пројекту, коришћене су различите технике тестирања како би се осигурала функционалност кључних делова система. Постман је популаран алат за тестирање API-ја који омогућава слање HTTP захтева и проверу одговора. Он је често коришћен за тестирање RESTfull API endpoint-a. Тестирање endpoint-а за добијање јела у ресторану:

GET localhost:8080/јеlo

Резултат теста:

[

{

"id": 1,

"naziv": "Pohovani kackavalj",

"kategorija": "predilo",

"cena": 690.0,

"opis": "Kockasti komadici kackavalja uvaljani u brasno, jaja i prezle. Przeni u vrelom ulju sve dok spoljasn"

},

{

"id": 2,

"naziv": "Tatarski biftek",

"kategorija": "predilo",

"cena": 2400.0,

"opis": "Sirovo parce juneceg bifteka sitno iseckano i pomesano sa zacinima: so, biber, seckani lik i kiseli "

},

{

"id": 3,

"naziv": "Grilovani losos sa limunom",

"kategorija": "glavno jelo",

"cena": 3399.0,

"opis": "Fileti lososa pripremljeni na rostilju zacinjeni solju, biberom, maslinovim uljem isvezim sokom limuna. Preporucena porcja je 200g po osobi."

},

{

"id": 4,

"naziv": "Teleci medaljon u sosu od pecuraka",

"kategorija": "glavno jelo",

"cena": 2989.0,

"opis": "Socni komadici mesa se kratko peku u tiganju da bi zadrzali socnost. Nakon pecenja se prelivaju soso"

},

{

"id": 5,

"naziv": "Cokoladni lava kolac ",

"kategorija": "desert",

"cena": 1255.0,

"opis": "Kolač se priprema od tamne čokolade, maslaca, jaja, šećera i brašna, a peče se kratko na visokoj temeraturi. Preporucena porcija je 250g po osobi."

},

{

"id": 6,

"naziv": "Panna cotta sa vocnim prelivom ",

"kategorija": "desert",

"cena": 1599.0,

"opis": "Panna cotta je kremasti italijanski desert napravljen od slatke pavlake, mleka, šećera, i želatina. "

},

Тестирање endpoint-а за добијање пића у ресторану:

GET localhost:8080/pice

[

{

"id": 1,

"naziv": "Turska kafa",

"cena": 160.0,

"tip": "topli napitak"

},

{

"id": 2,

"naziv": "Caj",

"cena": 100.0,

"tip": "topli napitak"

},

{

"id": 3,

"naziv": "Gazirana voda",

"cena": 120.0,

"tip": "bezalkoholno"

},

{

"id": 4,

"naziv": "Coca-cola",

"cena": 300.0,

"tip": "bezalkoholno"

},

{

"id": 5,

"naziv": "Jameson",

"cena": 500.0,

"tip": "alkoholno"

},

{

"id": 6,

"naziv": "Zajecarsko pivo",

"cena": 150.0,

"tip": "alkoholno"

}

]

# 4. Закључак

Пројекат пружа модеран, ефикасан и аутоматизован начин управљања кључним функцијама ресторана. Коришћењем Јаva Spring Boot frameworka и МySQL базе података, реализовали смо backen систем који омогућава једноставну интеграцију са клијентима путем REST АPI-ја. Кроз вишеслојну архитектуру пројекта (Controller, Service, Repository), осигурали смо модуларност и читљивост кода, омогућавајући даљу експанзију и одржавање система. Коришћењем Spring Boot JPA, омогућен је лак рад са базом података, док је употреба ЈSON формата осигурала једноставну комуникацију између клијента и сервера.

# Литература

[1.] Wikipedia

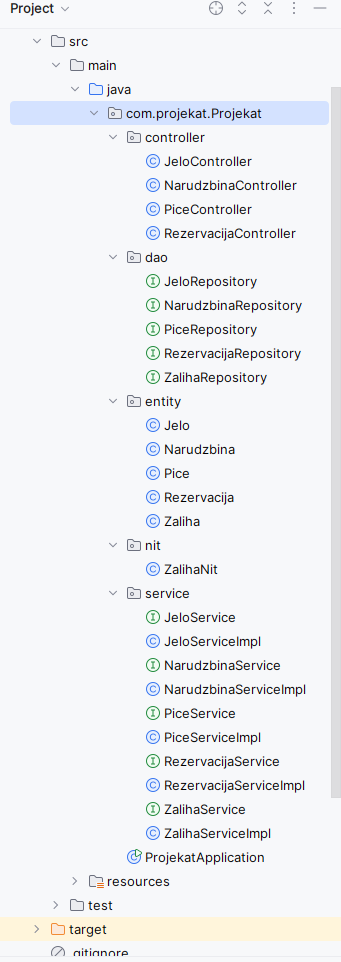
Презентације са предавања

[2.] Књига “Програмски језици” – Живковић М. и остали аутори

[3.] ChatGpt

# Прилог

Структура пројекта



pom.hml

*<?*xml version="1.0" encoding="UTF-8"*?>*<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">  
 <modelVersion>4.0.0</modelVersion>  
 <parent>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-parent</artifactId>  
 <version>3.3.3</version>  
 <relativePath/> *<!-- lookup parent from repository -->* </parent>  
 <groupId>com.projekat</groupId>  
 <artifactId>Projekat</artifactId>  
 <version>0.0.1-SNAPSHOT</version>  
 <name>Projekat</name>  
 <description>Demo project for Spring Boot</description>  
 <url/>  
 <licenses>  
 <license/>  
 </licenses>  
 <developers>  
 <developer/>  
 </developers>  
 <scm>  
 <connection/>  
 <developerConnection/>  
 <tag/>  
 <url/>  
 </scm>  
 <properties>  
 <java.version>17</java.version>  
 </properties>  
 <dependencies>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-data-jpa</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
 </dependency>  
  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-devtools</artifactId>  
 <scope>runtime</scope>  
 <optional>true</optional>  
 </dependency>  
 <dependency>  
 <groupId>com.mysql</groupId>  
 <artifactId>mysql-connector-j</artifactId>  
 <scope>runtime</scope>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-test</artifactId>  
 <scope>test</scope>  
 </dependency>  
  
 </dependencies>  
  
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-maven-plugin</artifactId>  
 </plugin>  
 </plugins>  
 </build>  
  
</project>

Јело Ентитет

package com.projekat.Projekat.entity;  
  
import jakarta.persistence.\*;  
  
@Entity  
@Table(name = "jelo")  
public class Jelo {  
  
 @Id  
 @GeneratedValue (strategy = GenerationType.*IDENTITY*)  
  
 @Column(name = "id")  
 private int id;  
  
 @Column(name = "naziv")  
 private String naziv;  
  
 @Column(name = "kategorija")  
 private String kategorija;  
  
 @Column(name = "cena")  
 private double cena;  
  
 @Column(name = "opis")  
 private String opis;  
  
 public Jelo() {  
 }  
  
 public Jelo(int id, String naziv, String kategorija, double cena, String opis) {  
 this.id = id;  
 this.naziv = naziv;  
 this.kategorija = kategorija;  
 this.cena = cena;  
 this.opis = opis;  
 }  
  
 public int getId() {  
 return id;  
 }  
  
 public void setId(int id) {  
 this.id = id;  
 }  
  
 public String getNaziv() {  
 return naziv;  
 }  
  
 public void setNaziv(String naziv) {  
 this.naziv = naziv;  
 }  
  
 public String getKategorija() {  
 return kategorija;  
 }  
  
 public void setKategorija(String kategorija) {  
 this.kategorija = kategorija;  
 }  
  
 public double getCena() {  
 return cena;  
 }  
  
 public void setCena(double cena) {  
 this.cena = cena;  
 }  
  
 public String getOpis() {  
 return opis;  
 }  
  
 public void setOpis(String opis) {  
 this.opis = opis;  
 }  
  
 @Override  
 public String toString() {  
 return "Jelo{" +  
 "id=" + id +  
 ", naziv='" + naziv + '\'' +  
 ", kategorija='" + kategorija + '\'' +  
 ", cena=" + cena +  
 ", opis='" + opis + '\'' +  
 '}';  
 }  
 public enum Kategorija {  
 *PREDJELO*, *GLAVNO\_JELO*, *DESERT* }  
}

Пиће Ентитет

package com.projekat.Projekat.entity;  
  
import jakarta.persistence.\*;  
  
@Entity  
@Table(name = "pice")  
public class Pice {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 @Column (name = "id")  
 private int id;  
  
 @Column(name = "naziv")  
 private String naziv;  
  
 @Column(name = "cena")  
 private double cena;  
  
 @Column(name = "tip")  
 private String tip;  
  
 public Pice() {  
 }  
  
 public Pice(int id, String naziv, double cena, String tip) {  
 this.id = id;  
 this.naziv = naziv;  
 this.cena = cena;  
 this.tip = tip;  
 }  
  
 public int getId() {  
 return id;  
 }  
  
 public void setId(int id) {  
 this.id = id;  
 }  
  
 public String getNaziv() {  
 return naziv;  
 }  
  
 public void setNaziv(String naziv) {  
 this.naziv = naziv;  
 }  
  
 public Double getCena() {  
 return cena;  
 }  
  
 public void setCena(Double cena) {  
 this.cena = cena;  
 }  
  
 public String getTip() {  
 return tip;  
 }  
  
 public void setTip(String tip) {  
 this.tip = tip;  
 }  
  
 @Override  
 public String toString() {  
 return "Pice{" +  
 "id=" + id +  
 ", naziv='" + naziv + '\'' +  
 ", cena=" + cena +  
 ", tip='" + tip + '\'' +  
 '}';  
 }  
   
 public enum Kategorija {  
 *TOPLI\_NAPITAK*, *BEZALKOHOLNO*, *ALKOHOLNO* }  
}

Резервација Ентитет

package com.projekat.Projekat.entity;  
  
import jakarta.persistence.\*;  
  
import java.sql.Time;  
import java.util.Date;  
  
@Entity  
@Table(name = " rezervacija")  
public class Rezervacija {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
  
 @Column(name = "id")  
 private int id;  
  
 @Column(name = "datum")  
 private Date datum;  
  
 @Column(name = "vreme")  
 private Time vreme;  
  
 @Column(name = "broj\_osoba")  
 private int broj\_osoba;  
  
 @Column(name = "kontakt\_info")  
 private String kontakt;  
  
 public Rezervacija() {  
 }  
  
 public Rezervacija(int id, Date datum, Time vreme, int broj\_osoba, String kontakt) {  
 this.id = id;  
 this.datum = datum;  
 this.vreme = vreme;  
 this.broj\_osoba = broj\_osoba;  
 this.kontakt = kontakt;  
 }  
  
 public int getId() {  
 return id;  
 }  
  
 public void setId(int id) {  
 this.id = id;  
 }  
  
 public Date getDatum() {  
 return datum;  
 }  
  
 public void setDatum(Date datum) {  
 this.datum = datum;  
 }  
  
 public Time getVreme() {  
 return vreme;  
 }  
  
 public void setVreme(Time vreme) {  
 this.vreme = vreme;  
 }  
  
 public int getBroj\_osoba() {  
 return broj\_osoba;  
 }  
  
 public void setBroj\_osoba(int broj\_osoba) {  
 this.broj\_osoba = broj\_osoba;  
 }  
  
 public String getKontakt() {  
 return kontakt;  
 }  
  
 public void setKontakt(String kontakt) {  
 this.kontakt = kontakt;  
 }  
  
 @Override  
 public String toString() {  
 return "Rezervacija{" +  
 "id=" + id +  
 ", datum=" + datum +  
 ", vreme=" + vreme +  
 ", broj\_osoba=" + broj\_osoba +  
 ", kontakt='" + kontakt + '\'' +  
 '}';  
 }  
}

Наручџбина Ентитет

package com.projekat.Projekat.entity;  
  
import jakarta.persistence.\*;  
  
@Entity  
@Table(name = "narudzbina")  
public class Narudzbina {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
  
 @Column(name = "id")  
 private int id;  
  
 @ManyToOne  
 @JoinColumn(name = "jelo\_id")  
 private Jelo jelo;  
  
 @ManyToOne  
 @JoinColumn(name = "pice\_id")  
 private Pice pice;  
  
 @Column(name = "kolicina\_jela")  
 private int kolicina\_jela;  
  
 @Column(name = "kolicina\_pica")  
 private int kolicina\_pica;  
  
 @Column(name = "ukupan\_iznos")  
 private double ukupan\_iznos;  
  
 @Column(name = "status\_narudzbine")  
 private StatusNarudzbine status;  
  
 public Narudzbina() {  
 }  
  
 public Narudzbina(int id, Jelo jelo, Pice pice, int kolicina\_jela, int kolicina\_pica, double ukupan\_iznos, StatusNarudzbine status) {  
 this.id = id;  
 this.jelo = jelo;  
 this.pice = pice;  
 this.kolicina\_jela = kolicina\_jela;  
 this.kolicina\_pica = kolicina\_pica;  
 this.ukupan\_iznos = ukupan\_iznos;  
 this.status = status;  
 }  
  
 public StatusNarudzbine getStatus() {  
 return status;  
 }  
  
 public void setStatus(StatusNarudzbine status) {  
 this.status = status;  
 }  
  
 public int getId() {  
 return id;  
 }  
  
 public void setId(int id) {  
 this.id = id;  
 }  
  
 public Jelo getJelo() {  
 return jelo;  
 }  
  
 public void setJelo(Jelo jelo) {  
 this.jelo = jelo;  
 }  
  
 public Pice getPice() {  
 return pice;  
 }  
  
 public void setPice(Pice pice) {  
 this.pice = pice;  
 }  
  
 public int getKolicina\_jela() {  
 return kolicina\_jela;  
 }  
  
 public void setKolicina\_jela(int kolicina\_jela) {  
 this.kolicina\_jela = kolicina\_jela;  
 }  
  
 public int getKolicina\_pica() {  
 return kolicina\_pica;  
 }  
  
 public void setKolicina\_pica(int kolicina\_pica) {  
 this.kolicina\_pica = kolicina\_pica;  
 }  
  
 public double getUkupan\_iznos() {  
 return ukupan\_iznos;  
 }  
  
 public void setUkupan\_iznos(double ukupan\_iznos) {  
 this.ukupan\_iznos = ukupan\_iznos;  
 }  
  
 @Override  
 public String toString() {  
 return "Narudzbina{" +  
 "id=" + id +  
 ", jelo=" + jelo +  
 ", pice=" + pice +  
 ", kolicina\_jela=" + kolicina\_jela +  
 ", kolicina\_pica=" + kolicina\_pica +  
 ", ukupan\_iznos=" + ukupan\_iznos +  
 ", status=" + status +  
 '}';  
 }  
  
 public enum StatusNarudzbine{  
 *na\_cekanju*, *u\_pripremi*, *dostavljeno* }  
}

Залиха Ентитет

package com.projekat.Projekat.entity;  
  
import jakarta.persistence.\*;  
  
@Entity  
@Table(name = "zalihe")  
public class Zaliha {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
  
 @Column(name = "id")  
 private int id;  
  
 @Column(name = "naziv")  
 private String naziv;  
  
 @Column(name = "kolicina")  
 private int kolicina;  
  
 public Zaliha() {  
 }  
  
 public Zaliha(int id, String naziv, int kolicina) {  
 this.id = id;  
 this.naziv = naziv;  
 this.kolicina = kolicina;  
 }  
  
 public int getId() {  
 return id;  
 }  
  
 public void setId(int id) {  
 this.id = id;  
 }  
  
 public String getNaziv() {  
 return naziv;  
 }  
  
 public void setNaziv(String naziv) {  
 this.naziv = naziv;  
 }  
  
 public int getKolicina() {  
 return kolicina;  
 }  
  
 public void setKolicina(int kolicina) {  
 this.kolicina = kolicina;  
 }  
  
 @Override  
 public String toString() {  
 return "Zalihe{" +  
 "id=" + id +  
 ", naziv='" + naziv + '\'' +  
 ", kolicina=" + kolicina +  
 '}';  
 }  
}

Залиха нит

package com.projekat.Projekat.nit;  
  
import com.projekat.Projekat.dao.ZalihaRepository;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.scheduling.annotation.Scheduled;  
import org.springframework.stereotype.Component;  
  
import java.io.FileWriter;  
import java.io.IOException;  
  
@Component  
  
public class ZalihaNit {  
  
 @Autowired  
 private ZalihaRepository zalihaRepository;  
  
 @Scheduled(fixedRate = 60000) *// Svakih 60 sekundi* public void proveriStanjeZaliha() {  
 zalihaRepository.findAll().forEach(zaliha -> {  
 if (zaliha.getKolicina() < 20) {  
 try (FileWriter writer = new FileWriter("obavestenja.txt", true)) {  
 writer.write("Niske zalihe za: " + zaliha.getNaziv() );  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
 });  
 }  
}

Jelo Repository

package com.projekat.Projekat.dao;  
  
import com.projekat.Projekat.entity.Jelo;  
import org.springframework.data.jpa.repository.JpaRepository;  
  
import java.util.Optional;  
  
public interface JeloRepository extends JpaRepository<Jelo, Integer> {  
  
  
 */\*List<Jelo> findByKategorija(String kategorija);  
 List<Jelo> svaJela();\*/* Optional<Jelo> findByNaziv(String naziv);  
 void deleteByNaziv(String naziv);  
}

Pice Repository

package com.projekat.Projekat.dao;  
  
import com.projekat.Projekat.entity.Pice;  
import org.springframework.data.jpa.repository.JpaRepository;  
  
import java.util.Optional;  
  
public interface PiceRepository extends JpaRepository<Pice, Integer> {  
  
  
 Optional<Pice> findByNaziv(String naziv);  
 void deleteByNaziv(String naziv);  
  
}

Rezervacija repository

package com.projekat.Projekat.dao;  
  
import com.projekat.Projekat.entity.Rezervacija;  
import org.springframework.data.jpa.repository.JpaRepository;  
  
public interface RezervacijaRepository extends JpaRepository<Rezervacija, Integer> {  
}

Narudzbina Repository

package com.projekat.Projekat.dao;  
  
import com.projekat.Projekat.entity.Narudzbina;  
import org.springframework.data.jpa.repository.JpaRepository;  
  
public interface NarudzbinaRepository extends JpaRepository<Narudzbina, Integer> {  
}

Zaliha Reposirory

package com.projekat.Projekat.dao;  
  
import com.projekat.Projekat.entity.Narudzbina;  
import org.springframework.data.jpa.repository.JpaRepository;  
  
public interface NarudzbinaRepository extends JpaRepository<Narudzbina, Integer> {  
}

Jelo Service

package com.projekat.Projekat.service;  
  
import com.projekat.Projekat.entity.Jelo;  
  
import java.util.List;  
  
public interface JeloService {  
  
  
 *//Jelo save(Jelo jelo);* List<Jelo> svaJela();  
  
 Jelo sacuvajJelo(Jelo jelo);  
  
 void obrisiJeloPoNazivu(String naziv);  
}

Pice Service

package com.projekat.Projekat.service;  
  
import com.projekat.Projekat.entity.Pice;  
  
import java.util.List;  
  
public interface PiceService {  
  
 List<Pice> svoPice();  
  
 Pice sacuvajPice(Pice pice);  
  
 void obrisiPicePoNazivu(String naziv);  
}

Rezervacija Service

package com.projekat.Projekat.service;  
  
import com.projekat.Projekat.entity.Rezervacija;  
  
import java.util.List;  
  
public interface RezervacijaService {  
  
 List<Rezervacija> sveRezervacije();  
 Rezervacija sacuvajRezervaciju(Rezervacija rezervacija);  
 void obrisiRezervacijuPoId(int id);  
}

Narudzbina Service

package com.projekat.Projekat.service;  
  
import com.projekat.Projekat.entity.Narudzbina;  
  
import java.util.List;  
  
public interface NarudzbinaService {  
  
 List<Narudzbina> sveNarudzbine();  
 Narudzbina sacuvajNarudzbinu(Narudzbina narudzbina);  
 void obrisiNarudzbinuPoId(int id);  
  
  
}

Zaliha Service

package com.projekat.Projekat.service;  
  
import com.projekat.Projekat.entity.Zaliha;  
  
import java.util.List;  
  
public interface ZalihaService {  
  
 List<Zaliha> sveZalihe();  
 Zaliha sacuvajZalihe(Zaliha zaliha);  
 void obrisiZalihuPoId(int id);  
}

Jelo ServiceImpl

package com.projekat.Projekat.service;  
  
import com.projekat.Projekat.entity.Jelo;  
import com.projekat.Projekat.dao.JeloRepository;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
  
import java.util.List;  
import java.util.Optional;  
  
@Service  
public class JeloServiceImpl implements JeloService {  
  
 @Autowired  
 private JeloRepository jeloRepository;  
  
 @Override  
 public List<Jelo> svaJela() {  
 return jeloRepository.findAll();  
 }  
  
 @Override  
 public Jelo sacuvajJelo(Jelo jelo) {  
 return jeloRepository.save(jelo);  
 }  
  
 @Override  
 public void obrisiJeloPoNazivu(String naziv) {  
 Optional<Jelo> jelo = jeloRepository.findByNaziv(naziv);  
 if (jelo.isPresent()) {  
 jeloRepository.deleteByNaziv(naziv);  
 } else {  
 throw new RuntimeException("Jelo sa nazivom " + naziv + " nije pronađeno.");  
 }  
 }  
  
  
}

Pice ServiceImpl

import com.projekat.Projekat.entity.Pice;  
import com.projekat.Projekat.dao.PiceRepository;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
  
import java.util.List;  
import java.util.Optional;  
  
@Service  
public class PiceServiceImpl implements PiceService {  
  
 @Autowired  
 private PiceRepository piceRepository;  
  
 @Override  
 public List<Pice> svoPice() {  
 return piceRepository.findAll();  
 }  
  
 @Override  
 public Pice sacuvajPice(Pice pice) {  
 return piceRepository.save(pice);  
 }  
  
 @Override  
 public void obrisiPicePoNazivu(String naziv) {  
 Optional<Pice> pice = piceRepository.findByNaziv(naziv);  
 if (pice.isPresent()) {  
 piceRepository.deleteByNaziv(naziv);  
 } else {  
 throw new RuntimeException("Pice sa nazivom " + naziv + " nije pronađeno.");  
 }  
 }  
}

Rezervacija ServiceImpl

package com.projekat.Projekat.service;  
  
import com.projekat.Projekat.entity.Rezervacija;  
import com.projekat.Projekat.dao.RezervacijaRepository;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
  
import java.util.List;  
import java.util.Optional;  
  
@Service  
public class RezervacijaServiceImpl implements RezervacijaService {  
  
 @Autowired  
 private RezervacijaRepository rezervacijaRepository;  
  
 @Override  
 public List<Rezervacija> sveRezervacije(){  
 return rezervacijaRepository.findAll();  
 }  
  
 @Override  
 public Rezervacija sacuvajRezervaciju(Rezervacija rezervacija) {  
 return rezervacijaRepository.save(rezervacija);  
 }  
  
 @Override  
 public void obrisiRezervacijuPoId(int id) {  
 Optional<Rezervacija> rezervacija = rezervacijaRepository.findById(id);  
 if (rezervacija.isPresent()) {  
 rezervacijaRepository.deleteById(id);  
 } else {  
 throw new RuntimeException("Rezervacija sa id-om " + id + " nije pronađeno.");  
 }  
 }  
  
}

Narudzbina ServiceImpl

package com.projekat.Projekat.service;  
  
import com.projekat.Projekat.entity.Narudzbina;  
  
import com.projekat.Projekat.dao.JeloRepository;  
import com.projekat.Projekat.dao.NarudzbinaRepository;  
import com.projekat.Projekat.dao.PiceRepository;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
  
import java.util.List;  
import java.util.Optional;  
  
@Service  
public class NarudzbinaServiceImpl implements NarudzbinaService {  
  
 @Autowired  
 private NarudzbinaRepository narudzbinaRepository;  
  
 @Autowired  
 private JeloRepository jeloRepository;  
  
 @Autowired  
 private PiceRepository piceRepository;  
  
 @Override  
 public List<Narudzbina> sveNarudzbine() {  
 return narudzbinaRepository.findAll();  
 }  
  
 @Override  
 public Narudzbina sacuvajNarudzbinu(Narudzbina narudzbina) {  
 */\* double cenaJela = narudzbina.getJelo().getCena() \* narudzbina.getKolicina\_jela();  
 double cenaPica = narudzbina.getPice().getCena() \* narudzbina.getKolicina\_pica();  
 narudzbina.setUkupan\_iznos(cenaJela + cenaPica);\*/* return narudzbinaRepository.save(narudzbina);  
 }  
 @Override  
 public void obrisiNarudzbinuPoId(int id) {  
 Optional<Narudzbina> narudzbina = narudzbinaRepository.findById(id);  
 if (narudzbina.isPresent()) {  
 narudzbinaRepository.deleteById(id);  
 } else {  
 throw new RuntimeException("Narudzbina sa id-om " + id + " nije pronađeno.");  
 }  
 }  
}

Zaliha ServiceImpl

package com.projekat.Projekat.service;  
  
import com.projekat.Projekat.entity.Zaliha;  
import com.projekat.Projekat.dao.ZalihaRepository;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
  
import java.util.List;  
import java.util.Optional;  
  
@Service  
public class ZalihaServiceImpl implements ZalihaService {  
  
 @Autowired  
 private ZalihaRepository zalihaRepository;  
  
 @Override  
 public List<Zaliha> sveZalihe() {  
 return zalihaRepository.findAll();  
 }  
  
 @Override  
 public Zaliha sacuvajZalihe(Zaliha zaliha ) {  
 return zalihaRepository.save(zaliha);  
  
 }  
  
 @Override  
 public void obrisiZalihuPoId(int id) {  
 Optional<Zaliha> rezervacija = zalihaRepository.findById(id);  
 if (rezervacija.isPresent()) {  
 zalihaRepository.deleteById(id);  
 } else {  
 throw new RuntimeException("Zaliha sa id-om " + id + " nije pronađeno.");  
 }  
 }  
}

Jelo Controller

package com.projekat.Projekat.controller;  
  
import com.projekat.Projekat.entity.Jelo;  
import com.projekat.Projekat.service.JeloService;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.web.bind.annotation.\*;  
  
import java.util.List;  
  
@RequestMapping("/jelo")  
@RestController  
public class JeloController {  
  
 @Autowired  
 private JeloService jeloService;  
  
 @GetMapping  
 public List<Jelo> svaJela() {  
 return jeloService.svaJela();  
 }  
  
 @PostMapping  
 public Jelo sacuvajJelo(@RequestBody Jelo jelo) {  
 return jeloService.sacuvajJelo(jelo);  
 }  
  
 @DeleteMapping("/јelo/{naziv}")  
 public void obrisiJeloPoNazivu(@PathVariable String naziv) {  
 jeloService.obrisiJeloPoNazivu(naziv);  
 }  
  
  
  
}

Pice Controller

package com.projekat.Projekat.controller;  
  
import com.projekat.Projekat.entity.Pice;  
import com.projekat.Projekat.service.PiceService;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.web.bind.annotation.\*;  
  
import java.util.List;  
  
@RequestMapping("/pice")  
@RestController  
public class PiceController {  
  
 @Autowired  
 private PiceService piceService;  
  
 @GetMapping  
 public List<Pice> svoPice(){  
 return piceService.svoPice();  
 }  
  
 @PostMapping  
 public Pice sacuvajPice( @RequestBody Pice pice) {  
 return piceService.sacuvajPice(pice);  
 }  
  
 @DeleteMapping("/pice/{naziv}")  
 public void obrisiPicePoNazivu(@PathVariable String naziv) {  
 piceService.obrisiPicePoNazivu(naziv);  
 }  
}

Rezervacija Controller

package com.projekat.Projekat.controller;  
  
import com.projekat.Projekat.entity.Rezervacija;  
import com.projekat.Projekat.service.RezervacijaService;  
  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.web.bind.annotation.\*;  
  
import java.util.List;  
  
@RequestMapping("/rezervacija")  
@RestController  
public class RezervacijaController {  
  
 @Autowired  
 private RezervacijaService rezervacijaService;  
  
 @GetMapping  
 public List<Rezervacija> sveRezervacije(){  
 return rezervacijaService.sveRezervacije();  
 }  
  
 @PostMapping  
 public Rezervacija sacuvajRezervaciju( @RequestBody Rezervacija rezervacija) {  
 return rezervacijaService.sacuvajRezervaciju(rezervacija);  
 }  
  
 @DeleteMapping("/rezervacija/{id}")  
 public void obrisiRezervacijuPoId(@PathVariable int id) {  
 rezervacijaService.obrisiRezervacijuPoId(id);  
 }  
}

Narudzbina Controller

package com.projekat.Projekat.controller;  
  
import com.projekat.Projekat.entity.Narudzbina;  
import com.projekat.Projekat.service.NarudzbinaService;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.web.bind.annotation.\*;  
  
import java.util.List;  
  
@RequestMapping("/narudzbina")  
@RestController  
public class NarudzbinaController {  
  
 @Autowired  
 private NarudzbinaService narudzbinaService;  
  
 @PostMapping  
 public List<Narudzbina> sveNarudzbine() {  
 return narudzbinaService.sveNarudzbine();  
 }  
  
 @GetMapping  
 public Narudzbina sacuvajNarudzbinu(@RequestBody Narudzbina narudzbina) {  
 return narudzbinaService.sacuvajNarudzbinu(narudzbina);  
 }  
  
 @DeleteMapping("/narudzbina/{id}")  
 public void obrisiNarudzbinuPoId(@PathVariable int id) {  
 narudzbinaService.obrisiNarudzbinuPoId(id);  
 }  
}

Zaliha Controller

package com.projekat.Projekat.controller;  
  
import com.projekat.Projekat.entity.Zaliha;  
import com.projekat.Projekat.service.ZalihaService;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.web.bind.annotation.\*;  
  
import java.util.List;  
  
@RestController  
@RequestMapping("/zaliha")  
public class ZalihaController {  
   
 @Autowired  
 private ZalihaService zalihaService;  
   
 @GetMapping  
 public List<Zaliha> sveZalihe() {  
 return zalihaService.sveZalihe();  
 }  
   
 @PostMapping  
 public Zaliha sacuvajZalihhu(@RequestBody Zaliha zaliha) {  
 return zalihaService.sacuvajZalihe(zaliha);  
 }  
   
 @DeleteMapping("/zaliha/{id}")  
 public void obrisiZalihuPoId(@PathVariable int id){  
 zalihaService.obrisiZalihuPoId(id);  
 }  
}