

Универзитет у Београду
ФАКУЛТЕТ ОРГАНИЗАЦИОНИХ НАУКА
Катедра за софтверско инжењерство

Семинарски рад из предмета Аутоматизација развоја софтвера

**Тема: Аутоматизација развоја софтверског система за
управљање административним пословима малопродајног
објекта „Бутик БИЛВА“**

Професор: Др Милош Милић **Студент:** Жељана Грујић 3703/2022

Београд, 2023.

1. Вербални опис

Неопходно је креирати апликацију за вођење евиденције о свим административним пословима малопродајног објекта „Бутик БИЛВА“, који се бави трговином одеће. Апликацијом је између осталог омогућено чување података о свим запосленима, купцима, добављачима, новим контактима, одећи, фактурама и ставкама фактуре за добављаче, као и рачуна и ставки рачуна за купце.

Регистрација новог корисника се заснива на уношењу података о имену, презимену, корисничком имену и лозинци. Кориснику може бити додељена улога основног корисника или запосленог.

Корисник чија је улога основни корисник може погледати понуду одевних комада који се могу пронаћи у објекту, док корисник чија је улога запослени добија више привилегија од којих: могућност да додели неком обичном кориснику улогу запослени и тиме му омогући све привилегије, унос нове локације и региона на коме се она налази, креирање и унос података о новим запосленима, контактима, добављачима и клијентима, креирање и ажурирање информација о називу и опису радног места, категорији, марки и моделу одеће, креирање и ажурирање одевног комада као и креирање и ажурирање фактуре за добављаче/купце и додавање више ставки наруџбине.

Креирање региона се заснива на уношењу података о називу региона, седишту тог округа и броја за локалну телефонску област. Креирање локације се заснива на одабиру региона на коме се налази локација, уношењу података о адреси, граду, и детаљима везаним за конкретну локацију (број улице, број зграде, број/спрат стана).

Креирање добављача, клијенента и контакта се заснива на уношењу података о имену, презимену, мејлу, контакт телефону, напомени/детаљима. За добављаче је потребно унети назив пословнице и локацију на којој се она налази.

Креирање новог типа радног места се заснива на уносу описног назива радног места и детаљима о наведеној позицији.

Креирање новог запосленог се заснива на уношењу података о имену, презимену, надимку, ЈМБГ-у, датуму рођења, контакт телефону, мејлу, датуму ангажовања, одабиру типа ангажовања, позицији на радном месту, адресе пребивалишта и додавањем слике запосленог. Како би запослени добио све привилегије система, мора му се поставити корисничко име.

Креирање нове категорије, марке, модела и статуса производа се заснива на уношењу података о називу и његовим детаљима.

Креирање новог производа се заснива на уношењу података о јединственом серијском коду, одабиру категорије, марке, модела, добављача и статуса, затим уноса доступне

количине, цене по комаду, датума куповине, додатних детаља, одговорног запосленог за пријем испоруке, и додавању слике модела.

Креирање фактуре се заснива на уносу датума, одабиру добављача/купца за кога се она креира, додавање детаља/напомене, додавању ставки фактуре.

Креирање ставке фактуре се заснива на избору производа и броја комада истог. Запослени може да креира више ставки наруџбине. Креирањем фактуре, поред осталог, памте се и све њене ставке.

2. Модел случајева коришћења

Могу се препознати следећи случајеви коришћења (Слика 1):

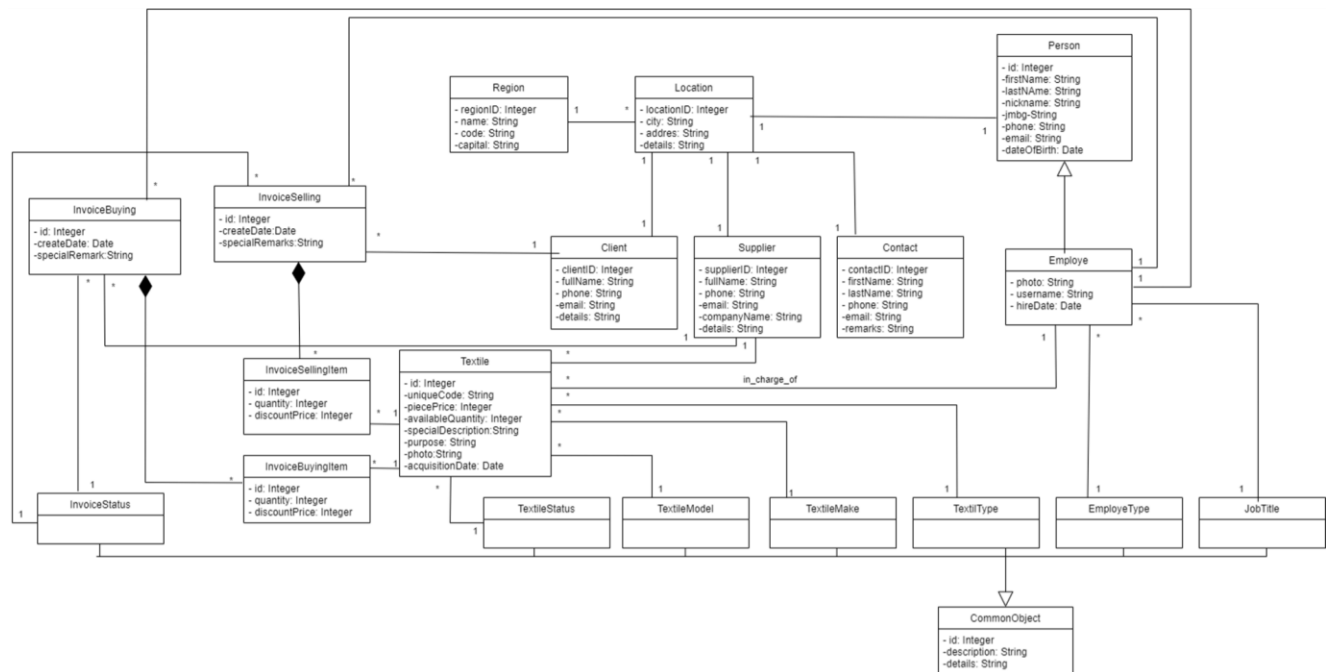
1. Унос региона
2. Измена региона
3. Брисање региона
4. Унос локације
5. Измена локације
6. Брисање локације
7. Унос запосленог
8. Измена запосленог
9. Брисање запосленог
10. Унос добављача
11. Измена добављача
12. Брисање добављача
13. Унос клијента
14. Измена клијента
15. Брисање клијента
16. Унос контакта
17. Измена контактаж
18. Брисање контакта
19. Унос радног места
20. Измена радног места
21. Брисање радног места
22. Унос радног ангажовања
23. Унос категорије производа
24. Измена категорије производа

25. Брисање категорије производа
26. Унос марке производа
27. Измена марке производа
28. Брисање марке производа
29. Унос модела производа
30. Измена модела производа
31. Брисање модела производа
32. Унос статуса производа
33. Унос одевног комада
34. Измена одевног комада
35. Брисање одевног комада
36. Унос фактуре за куповину-сложен случај коришћења
37. Измена фактуре за куповину-сложен случај коришћења
38. Унос фактуре за продају-сложен случај коришћења
39. Измена фактуре за продају-сложен случај коришћења
40. Унос статуса фактуре



Слика 1 Модел случајева коришћења

3. Концептуални модел- Дијаграм класа



Слика 2 Концептуални модел

4. Релациони модел

На основу концептуалног модела, добија се следећи релациони модел.

Region(RegionID,Name, Code, Capital)Location(LocationID, City, Address, Details, *CountryID*)

Contact(ContactID, FirstName, LastName, Email, Phone, Remarks, *LocationID*)

Client(ClientID, FullName, Email, Phone, Details, *LocationID*)

Supplier(SupplierID, FullName, Email, Phone, CompanyName, *LocationID*)

EmployeeType(EmployeeTypeID, Description, Details)

JobTitle(JobTitleID, Description, Details)

Employee(EmployeeID, FirstName, LastName, Nickname, JMBG, DateOfBirth, Email, Phone, Username, HireDate, *JobTitleID*, *EmployeeTypeID*, *LocationID*)

TextileType(TextileTypeID, Description, Details)

TextileMake(TextileMakeID, Description, Details)

TextileModel(TextileModelID, Description, Details)

TextileStatus(TextileStatusID, Description, Details)

Textile(TextileID, UniqueCode, PiecePrice, AvailableQuantity, SpecialDescription, Purpose, Photo, AcquisitionDate, *TextileTypeID*, *TextileMakeID*, *TextileModelID*, *TextileStatusID*, *SupplierID*, *EmployeeID*)

InvoiceStatus(InvoiceStatusID, Description, Details)

InvoiceBuying(InvoiceBuyingID, CreateDate, SpecialRemarks, *InvoiceStatusID*, *SupplierID*)

InvoiceBuyingItem(InvoiceBuyingItemID, Quantity, DiscountPrice, *InvoiceBuyingID*, *TextileID*)

InvoiceSelling(InvoiceSellingID, CreateDate, SpecialRemarks, *InvoiceStatusID*, *ClientID*)

InvoiceSellingItem(InvoiceSellingItemID, Quantity, DiscountPrice, *InvoiceSellingID*, *TextileID*)

Табела Region		Просто вредносно ограничење		Сложено вредносно ограничење		Структурно ограничење
Атрибути	Име	Тип атрибута	Вредност атрибута	Међузависност атрибута једне табеле	Међузависност атрибута више табела	INSERT / UPDATE CASCADES Location DELETE RESTRICTED Location
	RegionID	Integer	not null and > 0			
	Name	String	not null			
	Code	String	not null			
	Capital	String	not null			

Табела 1 Табела Region

Табела Location		Просто вредносно ограничење		Сложено вредносно ограничење		Структурно ограничење
Атрибути	Име	Тип атрибута	Вредност атрибута	Међузависност атрибута једне табеле	Међузависност атрибута више табела	INSERT RESTRICTED Region UPDATE RESTRICTED Region CASCADES Client, Supplier, Employee, DELETE RESTRICTED Client, Supplier, Employee
	LocationID	Integer	not null and > 0			
	City	String	not null			
	Address	String	not null			
	Details	String	not null			

Табела 2 Табела Location

Табела Contact		Просто вредносно ограничење		Сложено вредносно ограничење		Структурно ограничење
Атрибути	Име	Тип атрибута	Вредност атрибута	Међузависност атрибута једне табеле	Међузависност атрибута више табела	INSERT / UPDATE CASCADES / DELETE RESTRICTED /
	ContactID	Integer	not null and > 0			
	Firstname	String	not null			
	Lastname	String	not null			
	Email	String	not null			
	Phone	String	not null			
	Remarks	String				

Табела 3 Табела Contact

Табела Client		Просто вредносно ограничење		Сложено вредносно ограничење		Структурно ограничење
Атрибути	Име	Тип атрибута	Вредност атрибута	Међузависност атрибута једне табеле	Међузависност атрибута више табела	INSERT RESTRICTED Location UPDATE RESTRICTED Location CASCADES InvoiceSelling DELETE RESTRICTED InvoiceSelling
	ClientID	Integer	not null and > 0			
	Fullname	String	not null			
	Email	String	not null			
	Phone	String	not null			
	Details	String				

Табела 4 Табела Client

Табела Supplier		Просто вредносно ограничење		Сложено вредносно ограничење		Структурно ограничење
Атрибути	Име	Тип атрибута	Вредност атрибута	Међузависност атрибута једне табеле	Међузависност атрибута више табела	INSERT RESTRICTED Location UPDATE RESTRICTED Location CASCADES InvoiceBuying, Textile DELETE RESTRICTED InvoiceBuying, Textile
	SupplierID	Integer	not null and > 0			
	Fullname	String	not null			
	Email	String	not null			
	Phone	String	not null			
	CompanyName	String	not null			
	Details	String				InvoiceBuying, Textile

Табела 5 Табела Supplier

Табела EmployeeType		Просто вредносно ограничење		Сложено вредносно ограничење		Структурно ограничење
Атрибути	Име	Тип атрибута	Вредност атрибута	Међузависност атрибута једне табеле	Међузависност атрибута више табела	INSERT / UPDATE CASCADES Employee DELETE RESTRICTED Employee
	EmployeeTypeID	Integer	not null and > 0			
	Description	String	not null			
	Details	String	not null			

Табела 6 Табела EmployeeType

Табела JobTitle		Просто вредносно ограничење		Сложено вредносно ограничење		Структурно ограничење
Атрибути	Име	Тип атрибута	Вредност атрибута	Међузависност атрибута једне табеле	Међузависност атрибута више табела	INSERT / UPDATE CASCADES Employee DELETE RESTRICTED Employee
	JobStatusID	Integer	not null and > 0			
	Description	String	not null			
	Details	String	not null			

Табела 7 Табела JobTitle

Табела Employee		Просто вредносно ограничење		Сложено вредносно ограничење		Структурно ограничење
Атрибути	Име	Тип атрибута	Вредност атрибута	Међузависност атрибута једне табеле	Међузависност атрибута више табела	INSERT RESTRICTED JobTitle, EmployeeType, Location UPDATE RESTRICTED JobTitle, EmployeeTzpe, Location CASCADES Textile DELETE RESTRICTED Textile
	EmployeeID	Integer	not null and > 0			
	Firstname	String	not null			
	Lastname	String	not null			
	Nickname	String	not null			
	JMBG	String	not null			
	DateOfBirth	Date	not null			
	Email	String	not null			
	Phone	String	not null			
	Username	String	not null			
	HireDate	Date	not null			

Табела 8 Табела Employee

Табела TextileType		Просто вредносно ограничење		Сложено вредносно ограничење		Структурно ограничење
Атрибути	Име	Тип атрибута	Вредност атрибута	Међузависност атрибута једне табеле	Међузависност атрибута више табела	INSERT / UPDATE CASCADES Textile DELETE RESTRICTED Textile
	TextileTypeID	Integer	not null and > 0			
	Description	String	not null			
	Details	String	not null			

Табела 9 Табела TextileType

Табела TextileMake		Просто вредносно ограничење		Сложено вредносно ограничење		Структурно ограничење
Атрибути	Име	Тип атрибута	Вредност атрибута	Међузависност атрибута једне табеле	Међузависност атрибута више табела	INSERT / UPDATE CASCADES Textile DELETE RESTRICTED Textile
	TextileMakeID	Integer	not null and > 0			
	Description	String	not null			
	Details	String	not null			

Табела 10 Табела TextileMake

Табела TextileModel		Просто вредносно ограничење		Сложено вредносно ограничење		Структурно ограничење
Атрибути	Име	Тип атрибута	Вредност атрибута	Међузависност атрибута једне табеле	Међузависност атрибута више табела	INSERT / UPDATE CASCADES Textile DELETE RESTRICTED Textile
	TextileModelID	Integer	not null and > 0			
	Description	String	not null			
	Details	String	not null			

Табела 11 Табела TextileModel

Табела TextileStatus		Просто вредносно ограничење		Сложено вредносно ограничење		Структурно ограничење
Атрибути	Име	Тип атрибута	Вредност атрибута	Међузависност атрибута једне табеле	Међузависност атрибута више табела	INSERT / UPDATE
	TextileStatusID	Integer	not null and > 0			

	Description	String	not null			CASCADES Textile DELETE
	Details	String	not null			RESTRICTED Textile

Табела 12 Табела TextileStatus

Табела Textile		Просто вредносно ограничење		Сложено вредносно ограничење		Структурно ограничење
Атрибути	Име	Тип атрибута	Вредност атрибута	Међузависност атрибута једне табеле	Међузависност атрибута више табела	
	TextileID	Integer	not null and > 0			INSERT RESTRICTED TextileType, TextileMake, TextileModel, TextileStatus, Employee
	UniqueCode	String	not null			
	PiecePrice	String	not null and > 0			
	AvailableQuantity	String	not null and >= 0			UPDATE RESTRICTED TextileType, TextileMake, TextileModel, TextileStatus, Employee
	SpecialDescription	String	not null			
	Purpose	String	not null			CASCADES InvoiceBuyingItem, InvoiceSellingItem
	AcquisitionDate	Date	not null			
						DELETE RESTRICTED InvoiceBuyingItem, InvoiceSellingItem

Табела 13 Табела Textile

Табела InvoiceStatus		Просто вредносно ограничење		Сложено вредносно ограничење		Структурно ограничење
Атрибути	Име	Тип атрибута	Вредност атрибута	Међузависност атрибута једне табеле	Међузависност атрибута више табела	INSERT UPDATE CASCADES InvoiceBuying, InvoiceSelling DELETE RESTRICTED InvoiceBuying, InvoiceSelling
	InvoiceStatusID	Integer	not null and > 0			
	Description	String	not null			
	Details	String	not null			

Табела 14 Табела InvoiceStatus

Табела InvoiceBuying		Просто вредносно ограничење		Сложено вредносно ограничење		Структурно ограничење
Атрибути	Име	Тип атрибута	Вредност атрибута	Међузависност атрибута једне табеле	Међузависност атрибута више табела	INSERT RESTRICTED InvoiceStatus, Supplier UPDATE
	InvoiceBuyingID	Integer	not null and > 0			
	CreateDate	Date	not null			

	SpecialRemark	String	not null			RESTRICTED InvoiceStatus, Supplier CASCADES InvoiceBuyingItem DELETE RESTRICTED InvoiceBuyingItem
--	---------------	--------	----------	--	--	--

Табела 15 Табела InvoiceBuying

Табела InvoiceBuyingItem		Просто вредносно ограничење		Сложено вредносно ограничење		Структурно ограничење
Атрибути	Име	Тип атрибута	Вредност атрибута	Међузависност атрибута једне табеле	Међузависност атрибута више табела	INSERT RESTRICTED InvoiceBuying, Textile UPDATE RESTRICTED InvoiceBuying, Textile DELETE /
	InvoiceBuyingItemID	Integer	not null and > 0			
	Quantity	Integer	not null and > 0			
	DiscountPrice	Integer	not null and > 0			

Табела 16 Табела InvoiceBuyingItem

Табела InvoiceSelling		Просто вредносно ограничење		Сложено вредносно ограничење		Структурно ограничење
Атрибути	Име	Тип атрибута	Вредност атрибута	Међузависност атрибута једне табеле	Међузависност атрибута више табела	INSERT RESTRICTED InvoiceStatus, Client
	InvoiceSellingID	Integer	not null and > 0			

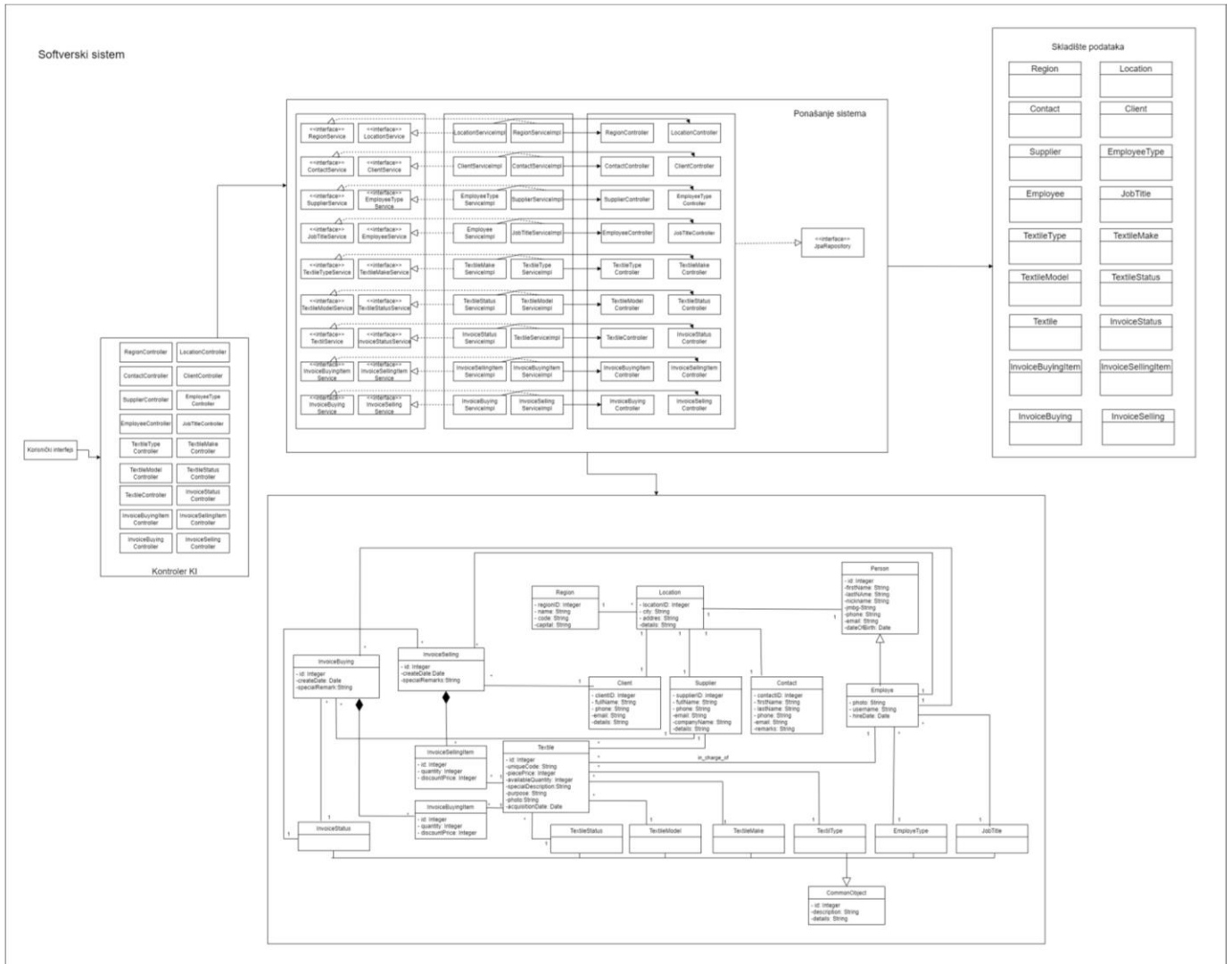
CreateDate	Date	not null			UPDATE RESTRICTED
SpecialRemark	String	not null			InvoiceStatus, Client CASCADES InvoiceSellingItem DELETE RESTRICTED InvoiceSellingItem

Табела 17 Табела InvoiceSelling

Табела InvoiceSellingItem		Просто вредносно ограничење		Сложено вредносно ограничење		Структурно ограничење
Атрибути	Име	Тип атрибута	Вредност атрибута	Међузависност атрибута једне табеле	Међузависност атрибута више табела	INSERT RESTRICTED InvoiceSelling, Textile UPDATE RESTRICTED InvoiceSelling, Textile DELETE /
	InvoiceSellingItemID	Integer	not null and > 0			
	Quantity	Integer	not null and > 0			
	DiscountPrice	Integer	not null and > 0			

Табела 18 Табела InvoiceSellingItem

5 Конечна архитектура



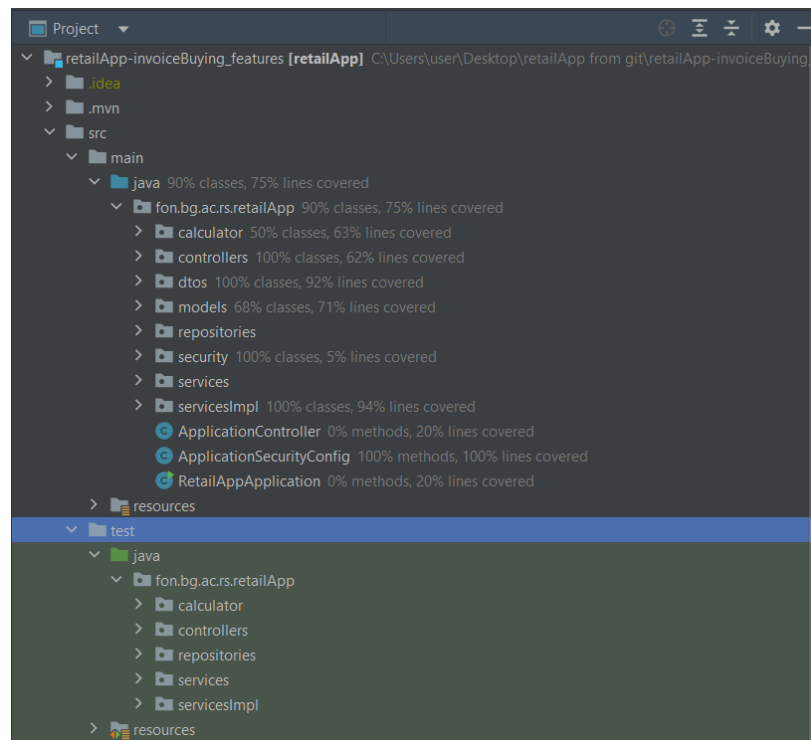
Слика 3 Коначна архитектура система

6 Тестирање софтвера

Како бисмо проверили како се софтверски систем понаша и да ли пружа очекивано понашање, написани су JUnit тестови који врше ове провере над коначним скупом случајева, након чега ћемо евалуирати резултате процеса тестирања.

Тестирање је реализовано комбинацијом мануелног и аутоматизованог притупа. Мануелним приступом је путем покретања корисничког интерфејса тестирана свака опција која је на располагању кориснику, а JUnit тестови преко којих је реализованом аутоматизовано тестирање заправо врше функционално тестирање и могу се покренути и искористити већи број пута. Како бисмо имали увид у то који део кода је покривен тестовима коришћена је JaCoCo библиотека која ће аутоматски израчунати која је покривеност кода тестовима, па тек када се изврши фаза тестирања JaCoCo извршава свој задатак који се тиче извештавања о тестовима. Иако треба тежити покривеношћу од 100%, ми ћемо се ипак задовољити са мањим процентом јер нема смисла писати тестове за класе из пакета: dtos и models. Како бисмо избегли конкретно везивање за податке и базу података и на тај начин створили зависност тестова и базе података, а такође и како бисмо избегли стварање зависности од метода које позивамо на нижим слојевима, а које смо притом већ тестирали на том нивоу (код контролера позивамо сервис класу која затим позива репозиторијум и тако комуницира са базом) коришћен је Mockito фрејмворк.

Након писања тестова структура пројекта је приказана на слици 4: Укупно је написано 313 тестова, где је у просеку за једну тест класу потребно да се изврши 20-30 секунди.



Слика 4 Структура пројекта

Ако погледамо JaCoCo извештај који добијамо након покретања свих тестова добијамо просечну покривеност тестовима од 52%. Одатле треба изузети поменуте пакете: models и dtos. Док је највећи значај придат пакетима controllers које су покривене тестовима са 64% и пакету servicesImpl који је покривен тестовима са 94%. Такође написани су тестови и за пакете: repository и services.

All in retailApp Coverage Results [Sessions](#)

All in retailApp Coverage Results

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
fon.bg.ac.rs.retailApp.models		39%		23%	541	813	30	187	86	341	1	22
fon.bg.ac.rs.retailApp.dtos		53%		28%	368	672	7	233	31	313	0	18
fon.bg.ac.rs.retailApp.controllers		64%		25%	33	134	195	527	14	114	0	20
fon.bg.ac.rs.retailApp.security.models		1%		0%	26	27	5	6	11	12	0	1
fon.bg.ac.rs.retailApp.security.controllers		2%		n/a	8	9	24	25	8	9	0	1
fon.bg.ac.rs.retailApp.security.servicesImpl		3%		0%	10	11	20	21	9	10	0	1
fon.bg.ac.rs.retailApp.servicesImpl		94%		0%	5	115	21	380	4	114	0	20
fon.bg.ac.rs.retailApp		86%		n/a	5	11	8	34	5	11	0	3
fon.bg.ac.rs.retailApp.calculator		65%		100%	2	5	5	12	2	4	1	2
Total	6,289 of 13,335	52%	1,300 of 1,738	25%	998	1,797	315	1,425	170	928	2	88

Created with JaCoCo 0.8.7.202105040129

Слика 5 Извештај тестирања

Ако погледамо детаљнији извештај везан за појединачну класу на пример TextileServiceImpl можемо видети да су све линије и гране кода покривене, као и све методе.

All in retailApp Coverage Results > fon.bg.ac.rs.retailApp.servicesImpl

fon.bg.ac.rs.retailApp.servicesImpl

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
TextileServiceImpl		100%		n/a	0	8	0	58	0	8	0	1
SupplierServiceImpl		100%		n/a	0	8	0	22	0	8	0	1

Слика 6 Извештај тестирања TextileServiceImpl класа

All in retailApp Coverage Results > fon.bg.ac.rs.retailApp.servicesImpl > TextileServiceImpl [Sessions](#)

TextileServiceImpl

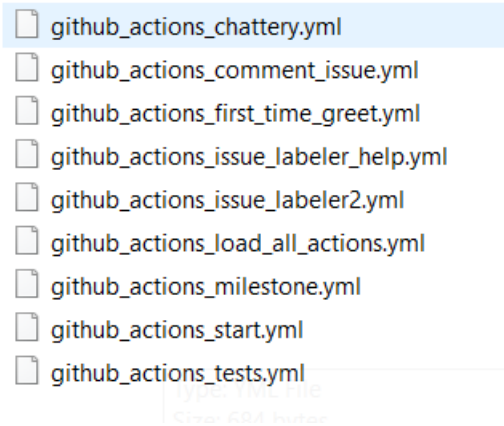
Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
lambda\$findByPurpose\$1(Textile)		100%		n/a	0	1	0	20	0	1
lambda\$get\$0(Textile)		100%		n/a	0	1	0	20	0	1
saveTextile(TextileDto)		100%		n/a	0	1	0	5	0	1
findById(int)		100%		n/a	0	1	0	4	0	1
findByPurpose(String)		100%		n/a	0	1	0	5	0	1
getTextiles()		100%		n/a	0	1	0	5	0	1
deleteById(int)		100%		n/a	0	1	0	2	0	1
TextileServiceImpl()		100%		n/a	0	1	0	1	0	1
Total	0 of 174	100%	0 of 0	n/a	0	8	0	58	0	8

Created with JaCoCo 0.8.7.202105040129

Слика 7 Извештај тестирања TextileServiceImpl класа

7 Верзионисање кода

Како бисмо испратили различите верзије и етапе кроз које апликација прошла, али такође и сачували све верзије кода на сигурном месту, а при томе и омогућили увид у историју рада и било који вид колаборације и сарадње са лакоћом, коришћена је GitHub платформа за верзионисање кода. Коришћење GitHub акција омогућава још један степен аутоматизације у току рада приликом развоја, тестирања, примене и испоруке софтвера. Git радни ток (workflow) омогућава аутоматизацију промена на инфраструктури. На следећим сликама можемо видети како је постигнута интеграција са GitHub акцијама које нам олакшавају континуирану интеграцију и континуирану испоруку. У оквиру једног тока (workflow) додате су следеће акције:



Слика 8 Креиране акције

Акција дефинисана `github_actions_start.yml` фајлом на сваку push промену која се деси над репозиторијумом треба да испише који догађај је узроковао њено покретање, помоћу ког оперативног система се извршава, на којој грани ког репозиторијума се налазимо, излистава све фајлове репозиторијума, статус извршења задатка као и све фајлове са `.yml` екстензијом који се односе на дефинисане акције тог пројекта.

```
name: GitHub-Actions-Start
on: [push]
jobs:
  Start:
    runs-on: ubuntu-latest
    steps:
      - run: echo "This is automatically triggered by ${github.event_name}"
      - run: echo "Used OS is ${runner.os}"
      - run: echo "We are on ${github.ref} branch, in repository ${github.repository}"
      - name: Checkout-Repository-Code
        uses: actions/checkout@v3
      - name: List all files in repo
        run: ls -l ${github.workspace}
      - run: echo "This job status is ${job.status}"
      - name: List YAML files in my workflows directory
        run: |
          find . -name '*.yml' -print
        working-directory: .github/workflows
```

Слика 9 github_actions_start.yml

Акција дефинисана github_actions_load_all_actions.yml фајлом на сваку push промену која се деси над репозиторијумом треба да учита све доступне акције дефинисане у action.yml фајлу, а затим да прикаже учитане акције. Резултат ће се приказати у оквиру низа акција ако постоје, ако не постоје низ ће бити празан.

```
name: GitHub-Actions-Load-All-Available-Actions
on: [push]
jobs:
  Load-all-available-actions:
    runs-on: ubuntu-latest
    steps:
      - name: Load available actions
        uses: devops-actions/load-available-actions@v2.0.0
        with:
          PAT: ${ secrets.GITHUB_TOKEN }
          user: ZeljanaGrujic
          id: load-actions
      - name: Display result file content
        run: |
          cat ${ steps.load-actions.outputs.actions-file-path }
      - run: echo "Used secret is ${ secrets.GITHUB_TOKEN }"
```

Слика 10 github_actions_load_all_actions.yml

Акција дефинисана github_actions_first_time_greet.yml фајлом на први направљени issue или pull_request корисника над нашим репозиторијумом аутоматски исписује обавештајну поруку за корисника.

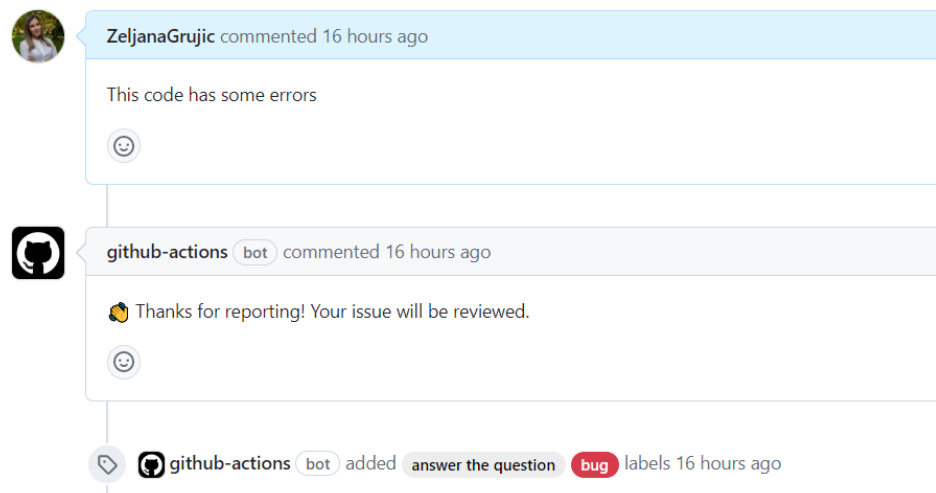
```
name: Greetings
on: [pull_request_target, issues]
jobs:
  send_first_greeting:
    runs-on: ubuntu-latest
    permissions:
      pull-requests: write
      issues: write
    steps:
      - uses: actions/first-interaction@v1
        with:
          repo-token: ${ secrets.GITHUB_TOKEN }
          issue-message: "Your issue will be reviewed soon."
          pr-message: "Thanks for your first pull request! Keep it up."
```

Слика 11 github_actions_first_time_greet.yml

Акција дефинисана `github_actions_comment_issue.yml` фајлом на сваки направљени issue над нашим репозиторијумом аутоматски испишује обавештајну поруку за корисника.

```
name: GitHub-Actions-Comment-Issue
on: [issues]
jobs:
  comment_issue:
    runs-on: ubuntu-latest
    permissions:
      issues: write
    steps:
      - uses: actions/github-script@v6.4.1
        with:
          script: |
            github.rest.issues.createComment({
              issue_number: context.issue.number,
              owner: context.repo.owner,
              repo: context.repo.repo,
              body: '🙏 Thanks for reporting! Your issue will be reviewed.'
            })
```

Слика 12 `github_actions_comment_issue.yml`



Слика 13 Након покретања `github_actions_comment_issue.yml`

Акције дефинисане `github_actions_issue_labeler_help.yml` и `github_actions_issue_labeler2.yml` фајловима на сваки направљени `issue` аутоматски стављају одговарајућу ознаку. Тако се на пример на покренути `issue` помоћу `github_actions_issue_labeler_help.yml` проверава да ли у наслову или у телу покренутог `issue` налази реч `help` и ако је то случај аутоматски се он означава ознакама `need_help` и `require_answer`.

```
name: Issue_Labeler_Help

on:
  issues:
    types:
      - reopened
      - opened
      - edited

jobs:
  apply-label:
    runs-on: ubuntu-latest
    permissions:
      issues: write
    steps:
      - name: Check if issue needs "need_help" label
        if: contains(github.event.issue.title, 'help') || contains(github.event.issue.body, 'help')
        uses: actions/github-script@v6
        with:
          script: |
            github.rest.issues.addLabels({
              issue_number: context.issue.number,
              owner: context.repo.owner,
              repo: context.repo.repo,
              labels: ["need_help", "require_answer"]
            })
```

Слика 14 `github_actions_issue_labeler_help.yml`

`Github_actions_issue_labeler2.yml` проверава да ли у наслову или у телу покренутог `issue` налази речи дефинисане у `labeler.yml` фајлу помоћу регуларних израза. Ако се налази реч `critical` или `urgent` додаје се `critical` ознака, ако се пронађу речи `not working` или `error` додаје се `bug` ознака, за `old version` или `update` се додаје `needs-updating` ознака, док се у свим осталим случајевима додаје `answer the question` ознака.

```
name: Issue-Labeler-Regex

on:
  issues:
    types: [opened, edited, reopened]

jobs:
  triage:
    runs-on: ubuntu-latest
    steps:
      - uses: github/issue-labeler@v3.1 #May not be the latest version
        with:
          repo-token: "${{ secrets.GITHUB_TOKEN }}"
          configuration-path: .github/labeler.yml
          enable-versioned-regex: 0
          include-title: 1
```

Слика 15 `github_actions_issue_labeler2.yml`

```
critical:
  - '(critical|urgent)'
bug:
  - '(not working|error)'
needs-updating:
  - '(old version|update)'
answer the question:
  - '/.*/'
```

Слика 16 github_actions_issue_labeler2.yml labeler.yml fajl

<input type="checkbox"/>	🕒 5 Open ✓ 21 Closed	Author ▾	Label ▾	Projects ▾	Milestones ▾	Assignee ▾	Sort ▾
<input type="checkbox"/>	🕒 This is critical #33 opened 1 hour ago by ZeljanaGrujic	answer the question	critical				💬 1
<input type="checkbox"/>	🕒 Need some help #32 opened 1 hour ago by ZeljanaGrujic	answer the question	need_help	require_answer			💬 1
<input type="checkbox"/>	🕒 Code error #28 opened 16 hours ago by ZeljanaGrujic	answer the question	bug				💬 1
<input type="checkbox"/>	🕒 Help test issue #27 opened yesterday by ZeljanaGrujic	answer the question	need_help	require_answer			💬 1
<input type="checkbox"/>	🕒 test issue #26 opened yesterday by ZeljanaGrujic						💬 1

Слика 17 Након покретања github_actions_issue_labeler2/help.yml

Акција дефинисана `github_actions_milestone.yml` фајлом сваког дана у 00.00 часова треба да дефинише контролну тачку, датум када треба урадити преглед достигнућа.

```
name: Weekly Milestones

on:
  schedule:
    - cron: 0 0 * * * # Run every Day at midnight

jobs:
  generate:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v2

      - name: Schedule Milestones
        uses: readmeio/scheduled-milestones@v1
        id: scheduled
        with:
          token: ${ secrets.GITHUB_TOKEN }
          title: Up for Review
          days: Tuesday,Thursday
          date_options: {"month": "long", "day": "numeric"}

      - name: Created Milestones
        run: echo ${ steps.scheduled.outputs.milestones }
```

Слика 15 `github_actions_milestone.yml`

Акција дефинисана `github_actions_chatterly.yml` фајлом за сваки покренути `pull_request` треба да направи линк путем кога ће бити могућа лична комуникација, размена аудио, видео и текстуалног садржаја порука. If услов је додат за случај да се акција не изврши успешно, да се она тада прескочи.

```
name: GitHub-Actions-Chatterly
on: [pull_request]
jobs:
  chatterly:
    name: chatterly
    runs-on: ubuntu-latest
    if: false
    steps:
      - run: echo "This is automatically triggered by ${github.event_name}"
      - name: post comment
        uses: lukejacksonn/chatterly@master
        env:
          GITHUB_TOKEN: ${ secrets.GITHUB_TOKEN }
```

Слика 16 `github_actions_chatterly.yml`

Акција дефинисана `github_actions_tests.yml` фајлом за сваки покренути `push` и `pull_request` над гранама `master` и `tests_feature` аутоматски покреће `build` пројекта и врши проверу да ли ће тај неки нови код који покушавамо да додамо да се, без проблема, уклопи у систем који већ имамо (на самом крају је додата команда којом се игнорише уколико се неки тест не изврши успешно).

```
name: Building project

on:
  push:
    branches: [ master, tests_feature ]
  pull_request:
    branches: [ master, tests_feature ]

jobs:
  build:
    runs-on: ubuntu-latest

    steps:
      - name: Checkout code
        uses: actions/checkout@v2

      - name: Set up JDK 12
        uses: actions/setup-java@v1
        with:
          java-version: '12'

      - name: Build with Maven
        run: mvn clean install
```

Слика 17 `github_actions_tests.yml`

8 Распоређивање софтвера

Обухвата све процесе, активности и кораке који се спроводе како би софтверски систем и/или ажурирање софтверског система било доступно, било корисницима било другим софтверским системима. Распоређивање софтвера се врши како би систем био доступан, док се доступност односи на степен у оквиру кога је систем/компонента система оперативна и доступна када је захтевано њено коришћење. У овом примеру је за распоређивање апликације коришћена Azure платформа. Постављена су следећа подешавања:

Create Web App ...

Instance Details

Need a database? [Try the new Web + Database experience.](#)

Name *

bilvaretailapp ✓
.azurewebsites.net

Publish *

☒ Code ☐ Docker Container ☐ Static Web App

Runtime stack *

Java 8 ▼

Java web server stack *

Java SE (Embedded Web Server) ▼

Operating System *

☒ Linux ☐ Windows

Region *

East US ▼
ⓘ Not finding your App Service Plan? Try a different region or select your App Service Environment.

Pricing plans

App Service plan pricing tier determines the location, features, cost and compute resources associated with your app.
[Learn more](#)

Linux Plan (East US) * ⓘ

(New) ASP-bilvaretailappgroup-aa50 ▼
[Create new](#)

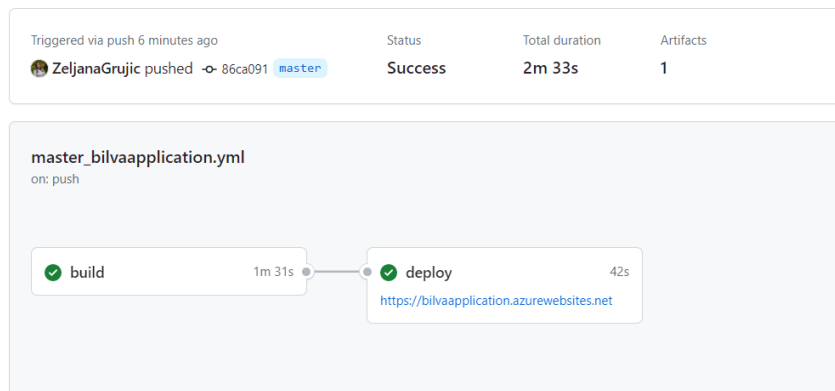
Pricing plan

Basic B1 (100 total ACU, 1.75 GB memory, 1 vCPU) ▼

Слика 18 Azure подешавања 1

Верзија јаве која је коришћена у пројекту је промењена са 12 на 8 јер Azure платформа има подршку за Јава 8 верзију, такође наша апликација ће бити доступна преко bilvaapplication.azurewebsites.net линка.

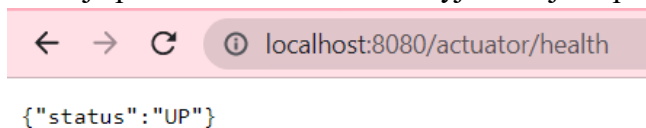
Након успешног постављања апликације она ће бити доступна преко наведеног линка, а још једна предност јесте да било које промене које направимо над апликацијом биће аутоматски проверене и интегрисане. Покретањем Git акција `tests_feature_java1.8_bilvaretailapp.yml` се врши аутоматско распоређивање апликације, приликом било које нове промене, на push догађај, овај фајл ће се поново аутоматски покретати, вршити проверу апликације а затим и распоређивање нове верзије. С обзиром да је апликација повезана на `localhost:3306/retailapp_db` нећемо моћи да је покренемо јер не можемо са web-а комуницирати са локалном базом података. Како би распоређивање било успешно, а поменути проблем са локалном базом био решен, коришћена је HSQLDB база података



Слика 20 GitHub акција коју креира Azure

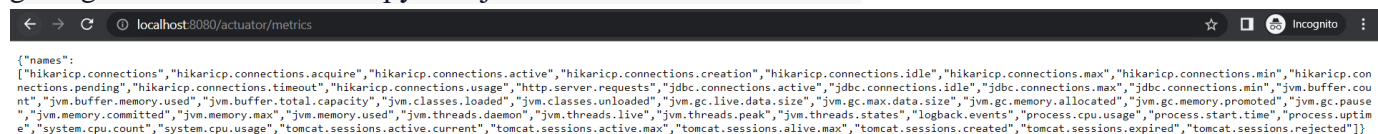
Проверу доступности наше апликације можемо вршити подешавањем Health Check опције на Azure платформи, или додавањем Spring Boot Actuator –а који ће нам омогућити да пратимо метрике доступности покренуте апликације путем приступних тачака. У наставку су приказане неке од приступних тачака кроз које се може пратити доступност апликације.

Кроз endpoint `/actuator/health` проверавамо да ли је наша апликација доступна, добијени статус UP означава да наша апликација ради како се од ње очекује и да је спремна да обрађује захтеве.



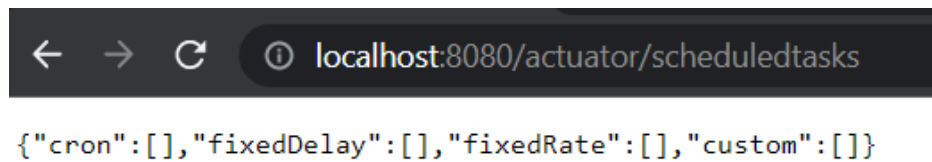
Слика 20 Spring Boot Actuator /health

Кроз endpoint `/actuator/metrics` добијамо увид у све метрике које користи наша Spring-Boot апликација. Тако на пример имамо метрику `http.server.requests` која показује број HTTP захтева добијених од сервера, `jvm.gc.memory.allocated` која показује колико меморије је одвојено за garbage collection и многе друге које можемо видети на слици.



Слика 21 Spring Boot Actuator /metrics

Кроз endpoint `/actuator/scheduledtasks` добијамо увид у планиране задатке за нашу апликацију. Кроз одговор који смо добили видимо да постоје три различита типа задатака који се могу заказати. Cron изразом специфицирамо време (секунде, минуте, сате, дане, месеце..) када задатак треба да се покрене, `fixedDelay` категорише задатке који треба да се понављају али са неким фиксним кашњењем између сваког покретања, `custom` категорише задатке који су заказани коришћењем `TaskScheduler` интерфејса.



Слика 22 Spring Boot Actuator `/scheduledtasks`

9 Паковање софтвера

Сам процес развоја софтвера је сложен, при чему се може користити велики број различитих радних окружења, технологија, алата, библиотека, сервиса. Сам поступак инсталације и конфигурације свих потребних зависности да би готов софтвер могао да се изврши, при чему треба имати у виду да треба да се омогући његово извршавање у више окружења, може бити веома захтеван, посебно ако се узме у обзир да се подешавања и упутства морају подесити мануелно. Овај поступак се може поједноставити коришћењем контејнера. Контејнер садржи зависности, конфигурацију, сервисе и њихову конфигурацију, који су потребни апликацији да би могла да се изврши. Када се све што је потребно апликацији, запакује у један контејнер, омогућава се једноставно извршавање у различитим окружењима помоћу једне команде, такође се омогућава и извршавање више верзија апликације без конфликта (јер свака верзија има свој изоловани контејнер). За креирање контејнера и паковање апликације у исти коришћена је Docker платформа. Како бисмо апликацију запаковали у изоловани контејнер, дефинисан је следећи Dockerfile.

```
Dockerfile > ...
1 FROM openjdk:8
2 RUN mkdir retailApp
3 ADD target/retailApp-0.0.1-SNAPSHOT.jar retailApp/retailApp-0.0.1-SNAPSHOT.jar
4 EXPOSE 8080
5 ENTRYPOINT ["java", "-jar", "retailApp/retailApp-0.0.1-SNAPSHOT.jar"]
```

Слика 23 Dockerfile

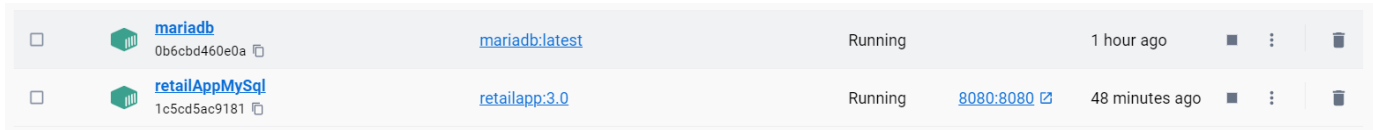
Дефинисана је и посебна мрежа која омогућава да се успостави комуникација између два изолована контејнера, једног на коме се налази и покреће наша апликација, и другог на коме се налази и покреће база података.









```
C:\Users\user>docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
89ab0e433ee7        bridge              bridge               local
22f1a27ac1c6        host                host                 local
47852c6cc94f        none                null                 local
1cbb4d96c5e1        retailApp-network   bridge               local

C:\Users\user>docker inspect retailApp-network
[
  {
    "Name": "retailApp-network",
    "Id": "1cbb4d96c5e103b69026091a37adcf598645ad147a9cfd9b878715c69e9c6664",
    "Created": "2023-04-29T11:41:32.802579212Z",
    "Scope": "local",
    "Driver": "bridge",
    "EnableIPv6": false,
    "IPAM": {
      "Driver": "default",
      "Options": {},
      "Config": [
        {
          "Subnet": "172.18.0.0/16",
          "Gateway": "172.18.0.1"
        }
      ]
    }
  }
],
```

Слика 24 RetailApp-Network

У Docker Desktop окружењу можемо видети два контејнера која међусобно комуницирају путем поменуте мреже. Њихова међусобна комуникација је могућа искључиво јер је дефинисана мрежа преко које они заједно комуницирају.



<input type="checkbox"/>	 mariadb 0b6cbd460e0a	mariadb:latest	Running	1 hour ago	  
<input type="checkbox"/>	 retailAppMySQL 1c5cd5ac9181	retailapp:3.0	Running	8080:8080 48 minutes ago	  

Слика 24 retailAppMySQL контејнер апликације комуницира са mariadb контејнером базе података

Како би се процес додатно аутоматизовао, и избегла комуникација два изолована контејнера преко заједничке мреже, могуће је извршити композицију и спојити оба контејнера у један. За то је потребно дефинисати docker-compose.yml фајл чији приказ је дат на наредној слици. Скрипта init.sql која садржи дефиницију шеме базе података, ће се аутоматски извршити приликом покретања контејнера. Именовани волумен db_data треба да омогући перзистентност података.

```
docker-compose.yml
1  version: '3.9'
2
3  services:
4    mariadb:
5      image: "mariadb"
6      container_name: "mariadb1"
7      ports:
8        - "3306:3306"
9      restart: unless-stopped
10     environment:
11       - MARIADB_USER=admin
12       - MARIADB_PASSWORD=admin
13       - MARIADB_ROOT_PASSWORD=root
14     command: --init-file /data/application/init.sql
15     volumes:
16       - ./init.sql:/data/application/init.sql
17       - db-data:/var/lib/mysql
18
19     app:
20       depends_on:
21         - "mariadb"
22       image: "retailapp:4.0"
23       container_name: "retailApp1"
24       ports:
25         - "8080:8080"
26       restart: on-failure
27
28     volumes:
29       db-data:
```

Слика 25 docker-compose.yml фајл

10 Закључак

Аутоматизација развоја софтвера и коришћење алата као што су Git, Azure и Docker постали су кључни за модерни софтверски инжењеринг. Ови алати омогућавају брз и ефикасан развој, тестирање, паковање и имплементирање софтверске апликације.

Код нашег студијског примера, коришћењем Git алата омогућено је праћење верзија кода, боља организација кода управљање променама насталим током развоја апликације. Такође је омогућено праћење историје промена у коду, што је корисно у случају потребе за повратком на претходну верзију.

Azure платформа пружа алате за аутоматску имплементацију апликација у облаку и управљање њима. Ово омогућава да се лакше скалира и управља апликацијама што знатно помаже приликом распоређивања софтвера.

Код нашег студијског примера, коришћењем Docker платформе омогућено је паковање апликације у контејнере који се могу лако преносити и извршавати на различитим окружењима, без претходне потребе за сложеним конфигурисањем.

Коришћење ових алата и аутоматизација развоја софтвера у целини, убрзава животни циклус развоја софтвера и смањује број грешака у коду. Аутоматизација развоја софтвера применом ових алата ствара могућност за бржу и ефикаснију испоруку производа на тржиште, што је кључно у данашњем конкурентном окружењу.