



Prolećni semestar, 2022/23

PREDMET: IT355 Veb Sistemi 2

Projektni zadatak

Onlajn prodavnica video igara

Student: Bojana Stajić 4596

Profesor : Vladimir Milićević

Asistent: Miloš Milašinović

Sadržaj projekta

Specifikacija softverskog sistema	3
Kratak opis poslovnog problema koji se rešava softverskim sistemom i njegova veza sa okruženjem	3
Tehničke specifikacije	3
Funkcionalni zahtevi softverskog sistema	3
Nefunkcionalni zahtevi softverskog sistema	4
Dijagram arhitekture	4
Slučajevi korišćenja sistema	6
Prikaz interfejsa i demonstracija njegovog rada	7
<i>Entiteska klasa</i>	7
<i>Servisna klasa</i>	7
<i>Repository klasa</i>	9
<i>HTML stranica kod</i>	9
<i>Screenshot-ovi i rad projekta</i>	10
Testiranje	17
Upotreba Aktuatora	20
/actuator	20
/health	20
/beans	21
Prikaz aktuatora unutar razvojnog okruženja	21
Zaključak	23
Source code	23
Literatura	23

Specifikacija softverskog sistema

Kratak opis poslovnog problema koji se rešava softverskim sistemom i njegova veza sa okruženjem

Aplikacija koja je predlog za ovaj projektni zadatak jeste Onlajn prodavnica video igara koja implementira sve potrebne stavke iz uputstva za projekat. Naime, aplikacija jeste jedan Maven Spring Boot projekat koji se sastoji od registracije/login-a, a zatim i mogućnosti kupovine video igara iz ponude, dodaja istih u korpu kao i realizacija same kupovine. Backend aplikacije će biti realizovan putem MySQL baze podataka. Drugi tip korisnika jeste admin koji ima uvid u postojeću ponudu igrica i aktivnih korisnika. Admin može da doda, obriše ili izmeni igrice iz ponude i da obriše korisnike. Ideja je da se kreira aplikacija koja poštuje osnovne Spring koncepte, kao i jednostavnost upotrebe kod ovakvih sistema kreirajući jednostavnu aplikaciju za rad.

Tehničke specifikacije

Korisniku je potreban najobičniji računar neke normalne snage, sa minimalnim zahtevima kao što su:

CPU: 1 gigahertz (GHz) ili jače sa dva ili više jezgra na kompatibilnom 64-bitnom procesoru.

RAM: 4 gigabytes (GB) ili jače

GPU: bilo koja kompatibilna sa DirectX 12 ili kasnije

Internet konekcija: Internet konekcija je neophodna da bi se pristupilo sistemu s obzirom na to da je ovo web aplikacija. Koristiti neki od najnovijih verzija pretraživača (Google, Safari, Opera, Mozilla)

Tehnologije koje su korišćene prilikom izrade aplikacije su sledeće: Spring Boot, Thymeleaf, MySQL, BootstrapCSS, Maven.

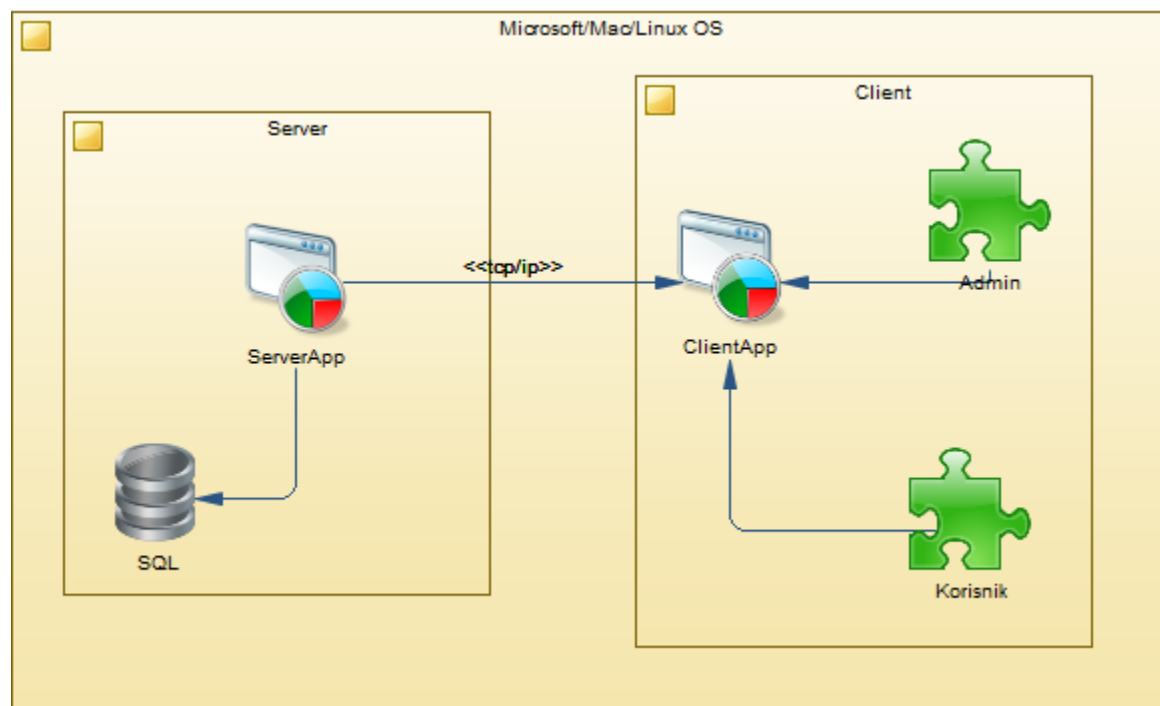
Funkcionalni zahtevi softverskog sistema

Title ID	Full Description	Code	Priority	Workload	Risk	Status
1.	Korisnik	REQ_0001	Undefined	0	Undefined	Draft
1.1	Registracija Korisnik ima mogućnost registracije na sistem ukoliko nema profil ili mu pristupa prvi put.	REQ_0002	Undefined		Undefined	Draft
1.2	Login Korisnik ima mogućnost prijave na sistem sa validnim postojećim podacima.	REQ_0003	Undefined		Undefined	Draft
1.3	Pregled ponude Korisnik ima mogućnost pregleda postojeće ponude prodavnice zajedno sa svojim podacima.	REQ_0004	Undefined	0	Undefined	Draft
1.3.1	Detaljan pregled proizvoda Korisnik ima detaljno pregleda odabranog proizvoda iz ponude prodavnice na zasebnoj stranici sa opisanim detaljima.	REQ_0005	Undefined		Undefined	Draft
1.3.2	Odabir proizvoda Korisnik ima opciju kupovine odabranog proizvoda i time ga dodaje u korpu.	REQ_0006	Undefined	0	Undefined	Draft
1.3.2.	Pregled korpe Korisnik ima uvid u proizvode dodate u korpu kao i mogućnost izmene količine odabranog proizvoda.	REQ_0007	Undefined		Undefined	Draft
1.3.2.	Realizacija kupovine Korisnik potvrđuje kupovinu i dobija poruku o uspešnosti svoje porudžbine.	REQ_0008	Undefined		Undefined	Draft
2.	Administrator	REQ_0009	Undefined	0	Undefined	Draft
2.1	Login Admin ima mogućnost prijave na sistem sa svojim podacima.	REQ_0010	Undefined		Undefined	Draft
2.2	Upravljanje podacima Admin ima mogućnost upravljanja podacima o proizvodima(dodaj,obriši,izmeni) i o korisnicima(obriši).	REQ_0011	Undefined		Undefined	Draft

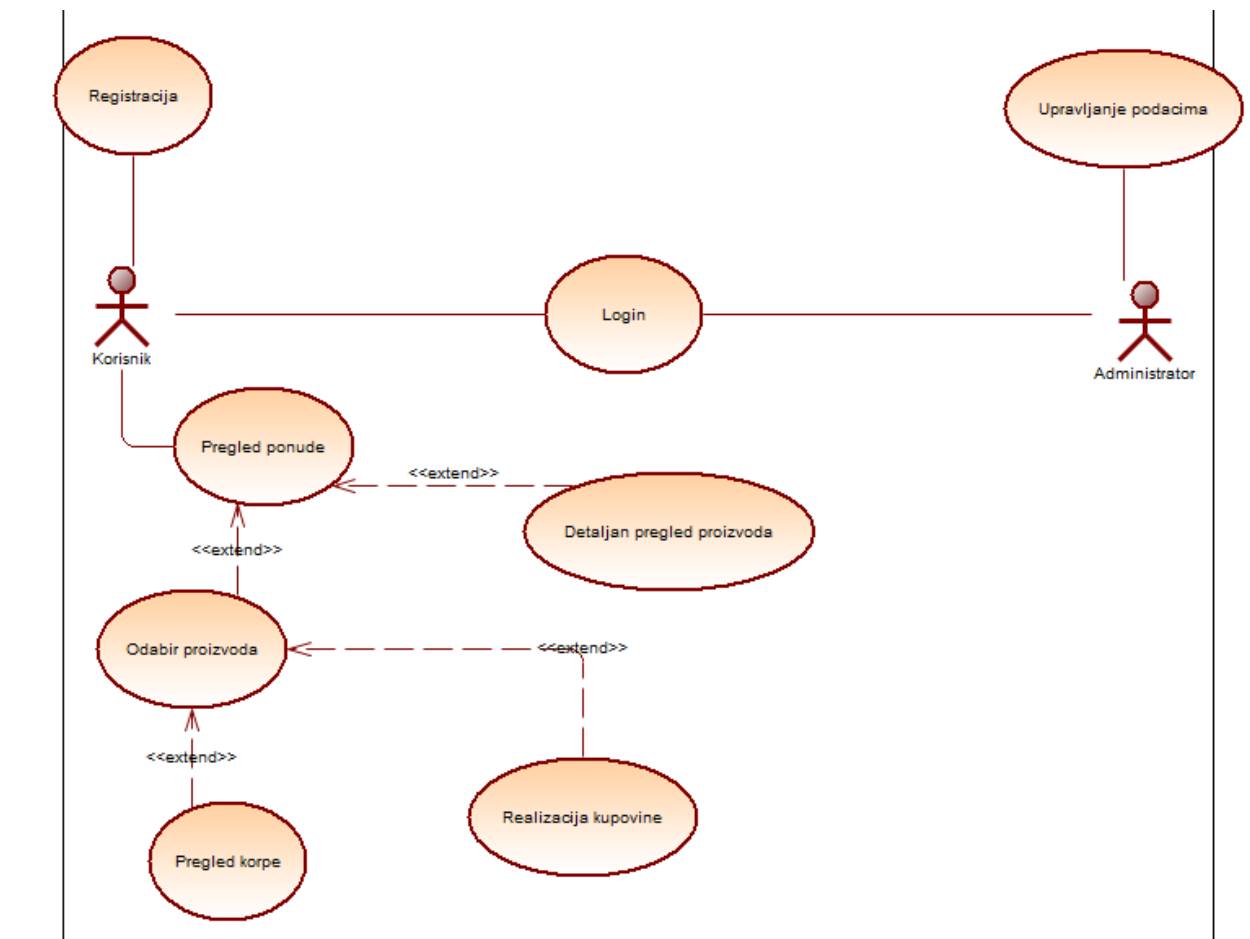
Nefunkcionalni zahtevi softverskog sistema

	Title ID	Full Description	Code	Priority	Workload	Risk	Status
→	1.	Lakoća upotrebe Sistem treba da bude intuitivan i lak za upotrebu korisnicima različitog stepena pismenosti iz oblasti IT-a.	REQ_0001	1		Low	Draft
2	2.	Brzina Sistem treba da ima brz odziv na korisničke zahteve i da nudi optimalna rešenja.	REQ_0002	1		Undefined	Draft
3	3.	Bezbednost Sistem treba da garantuje zaštitu podataka korisnika, kao i osetljivih informacija u svakom datom trenutku.	REQ_0003	Undefined		High	Draft
4	4.	Raspoloživost Sistem treba da bude dostupan korisnicima 90% vremena na mesečnom nivou.	REQ_0004	1		Low	Draft

Dijagram arhitekture



Slučajevi korišćenja sistema



Prikaz interfejsa i demonstracija njegovog rada

Entitetska klasa

```
@Entity
@Table(name = "users", uniqueConstraints = @UniqueConstraint(columnNames = "email"))
public class User {

    2 usages
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;

    3 usages
    @Column(name = "first_name")
    private String firstName;

    3 usages
    @Column(name = "last_name")
    private String lastName;

    3 usages
    @Column(name = "email")
    private String email;

    3 usages
    @Column(name = "password")
```

*Slika 1 – Primer entitetske klase User

Servisna klasa

```

@Service
public class UserServiceImpl implements UserService{

    3 usages
    private UserRepository userRepository;

    1 usage
    @Autowired
    private BCryptPasswordEncoder passwordEncoder;

    public UserServiceImpl(UserRepository userRepository) {
        super();
        this.userRepository = userRepository;
    }

    1 usage
    @Override
    public User save(UserRegistrationDto registrationDto) {
        User user = new User(registrationDto.getFirst_name(),
            registrationDto.getLast_name(), registrationDto.getEmail(),
            passwordEncoder.encode(registrationDto.getPassword()), Arrays.asList(new Role( name: "ROLE_USER"));

        return userRepository.save(user);
    }
}

```

*Slika 2 – UserServiceImpl klasa iz service paketa

```

6 usages 1 implementation
public interface UserService extends UserDetailsService{

    1 usage 1 implementation
    User save(UserRegistrationDto registrationDto);

}

```

*Slika 3 – UserService klasa iz service paketa

Repository klasa

```
2 usages
public interface GameRepository extends CrudRepository<Game, Long> {
    4 usages
    Optional<Game> findById(Long id);
}
```

*Slika 4 – GameRepository klasa sa metodom za pronalazak igre preko njegov id-ja

HTML stranica kod

```
<br>
<br>
<div class = "container">
  <div class = "row">
    <div class = "col-md-6 col-md-offset-3">

      <h1> Bojana's Place Games</h1>
      <h1> Login Page </h1>
      <form th:action="@{/login}" method="post">

        <!-- error message -->
        <div th:if="${param.error}">
          <div class="alert alert-danger" th:text="#{invalidUser}"></div>
        </div>

        <!-- logout message -->
        <div th:if="${param.logout}">
          <div class="alert alert-info" th:text="#{logoutUser}"></div>
        </div>

        <div class = "form-group">
          <label for = "username" th:text="#{username}"/>:
          <input type="text" class = "form-control" id = "username" name = "username"
            placeholder="Enter Email" autofocus="autofocus">

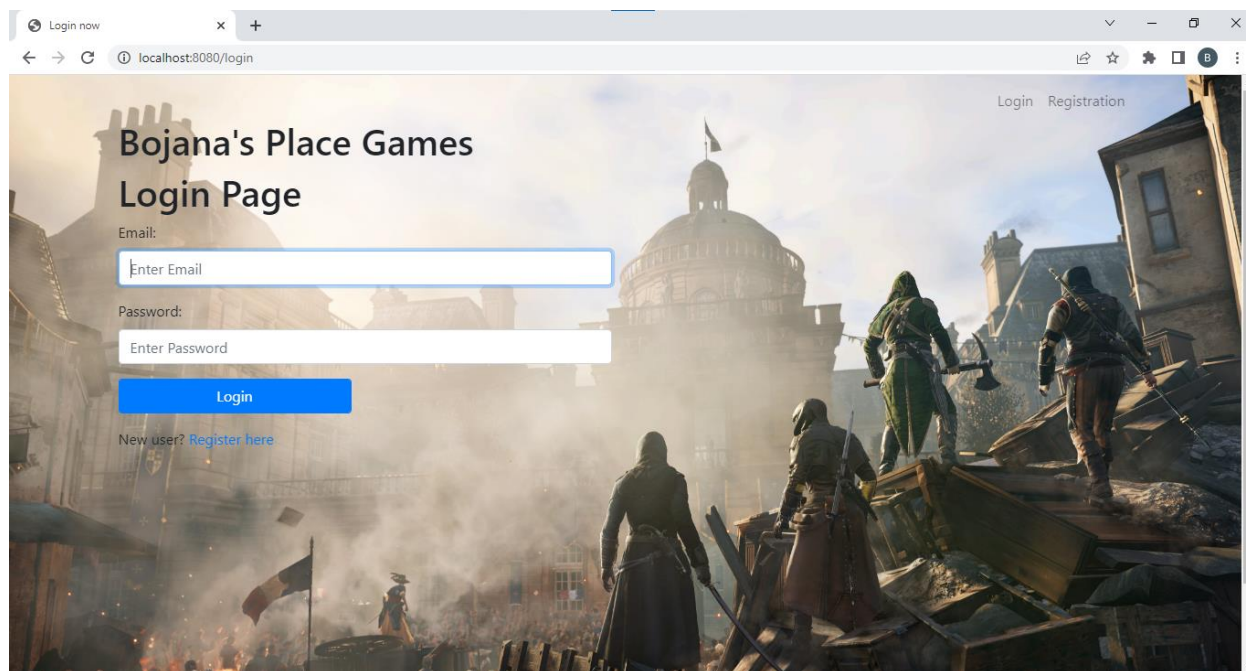
```

*Slika 5 – Deo koda HTML stranice za login korisnika

-Sve prethodne slike u okviru ovog poglavlja su prikazane kao primer samo jedne klase ili stranice svog odgovarajućeg segmenta u projektu, ali kako njih ima više, a identične su u smislu strukture, nisu sve slikane da ne bi bilo dupliranih i redundantnih podataka u dokumentu.

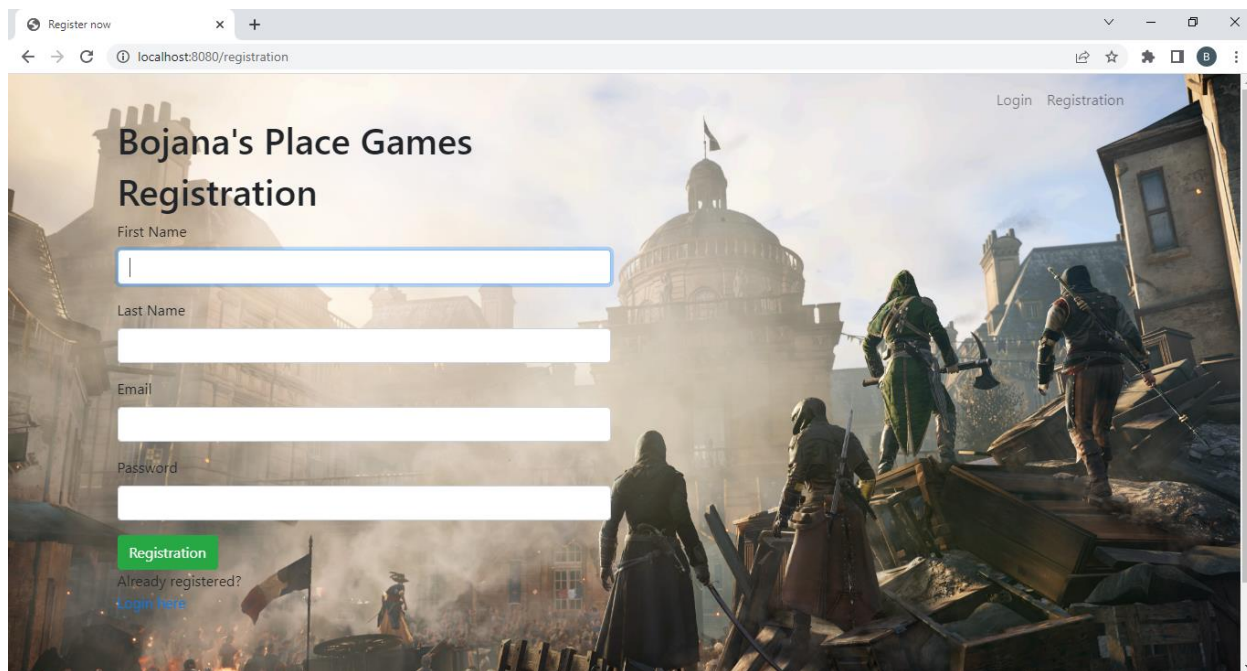
Screenshot-ovi i rad projekta

Login stranica



*Slika 6 – Login stranica

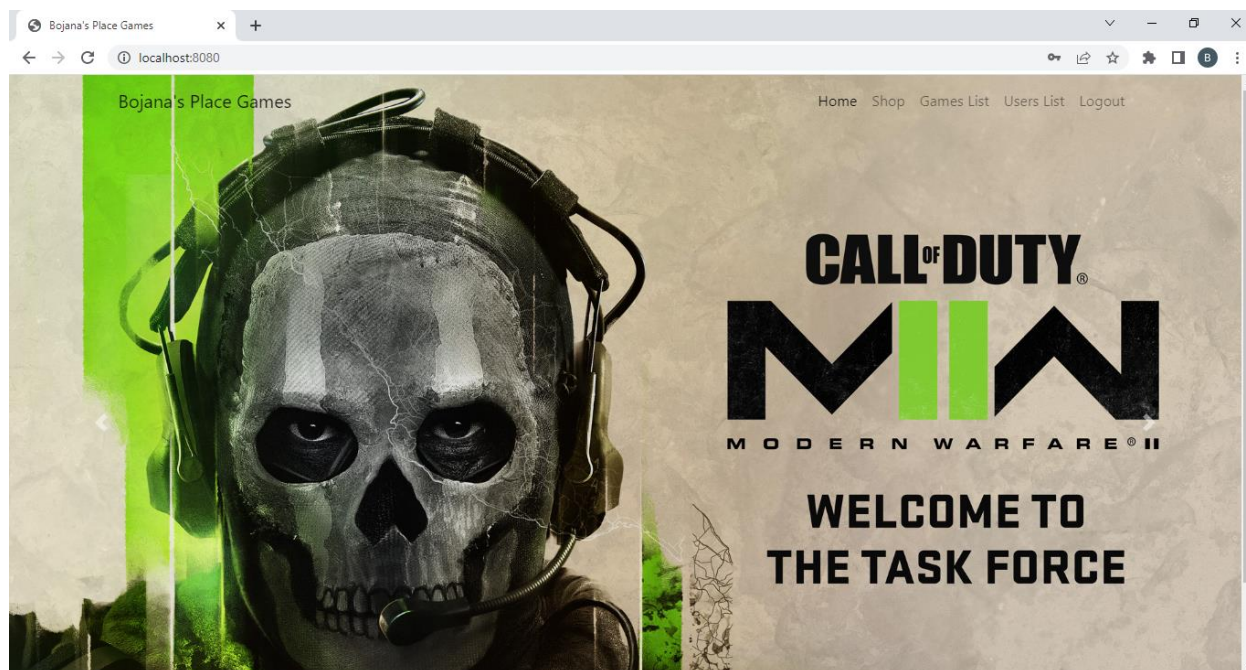
Registration stranica



*Slika 7 – Registration stranica

Na prethodne dve slike prikazane su početna stranica projekta (login page) i register stranica ukoliko je novi korisnik u pitanju. Kada korisnik napravi profil uspešno, te podatke unosi u login formu i pristupa početnoj/home stranici aplikacije. Funkcionalnosti se razlikuju na osnovu role-a korisnika, odnosno da li je on admin ili ne. Prilikom registracije default uloga je običan korisnik.

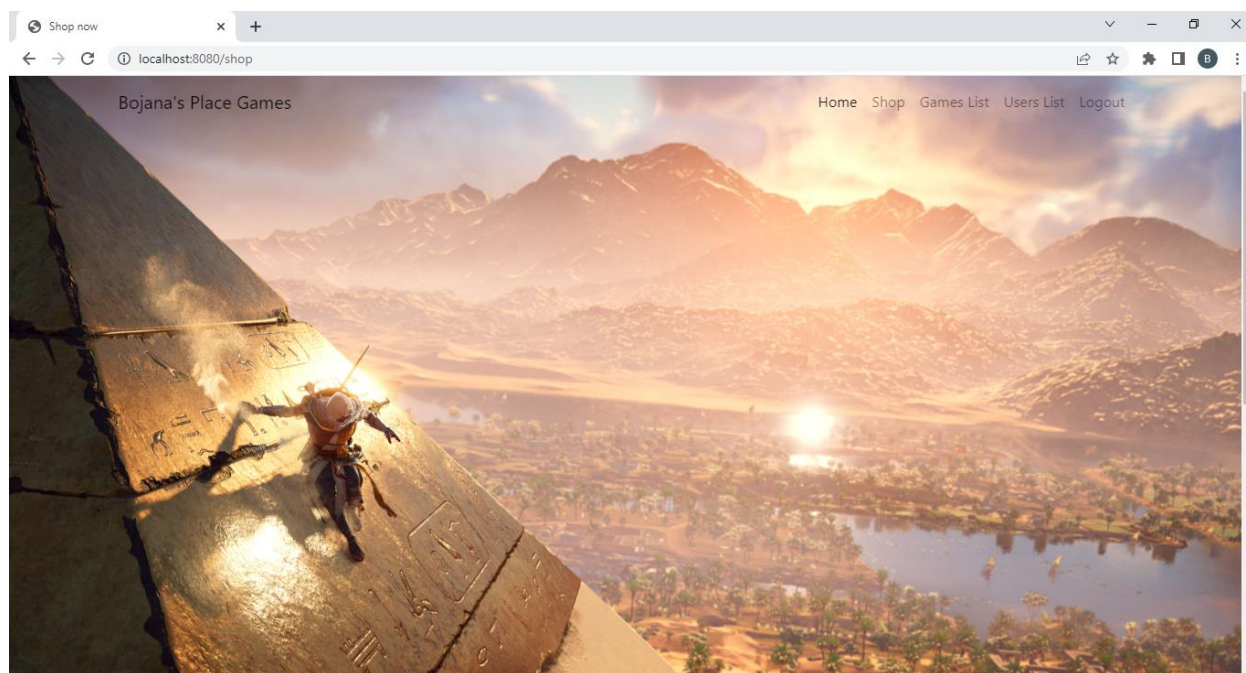
Početna stranica (običan korisnik i admin)



*Slika 8 – Izgled početne stranice sa slider-om koji prikazuje aktuelne igrice

Shop stranica (Kupovina)

Ovu funkcionalnost dele i admin i korisnik, te ćemo zbog dupliranja podataka prikazati kako to izgleda kod običnog korisnika, a kod admina je svakako isto.

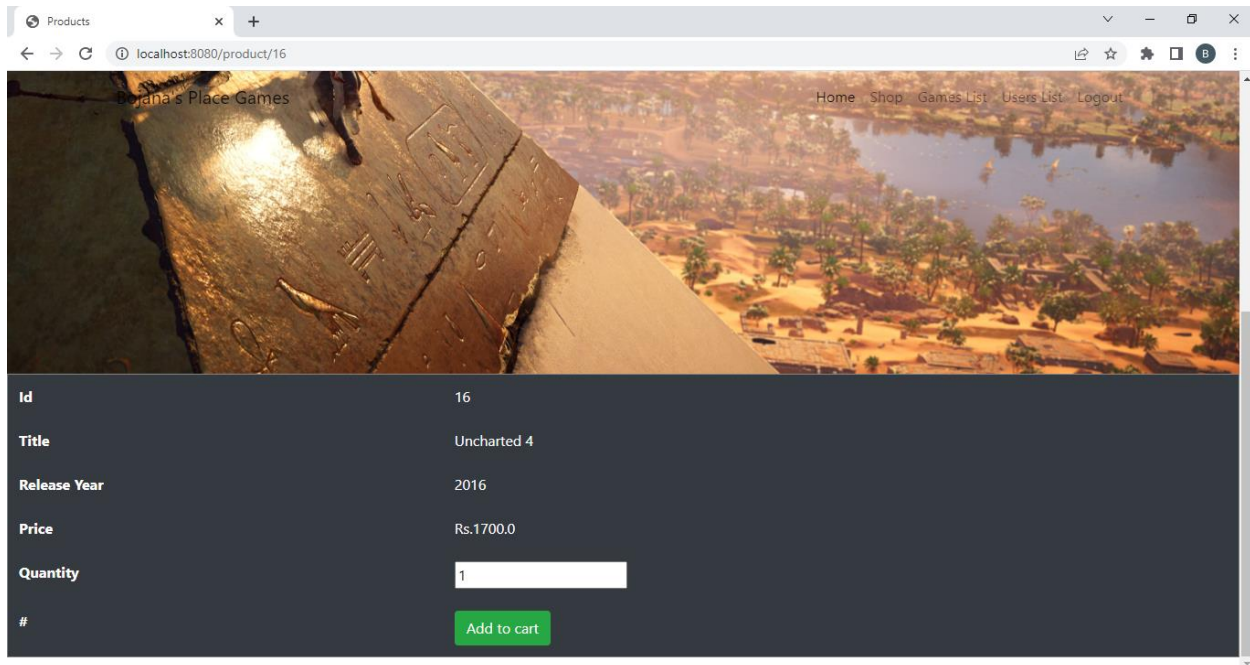


*Slika 9 – Izgled i slika stranice za kupovinu, a kada scroll-ujemo dole dolazimo do ponude

A screenshot of the same web browser, but scrolled down to show a list of games for sale. The list is presented in a table format with columns for 'Id', 'Title', 'Release Year', 'Genre', 'Quantity', 'Price', and '#'. Each row represents a different game, and each row has an 'Add to cart' button. The background of the page is a dark, textured surface, possibly a game cover or a solid color.

Id	Title	Release Year	Genre	Quantity	Price	#
3	Assassins Creed Valhalla	2020	RPG	5	2500.0	Add to cart
4	Watch Dogs 2	2016	Adventure	3	1700.0	Add to cart
5	Call of Duty Modern Warfare	2019	First person shooter	6	2000.0	Add to cart
6	GTA 5	2013	Open world	7	1800.0	Add to cart
7	Call of Duty Modern Warfare 2	2022	First person shooter	9	4500.0	Add to cart
16	Uncharted 4	2016	Adventure	3	1700.0	Add to cart

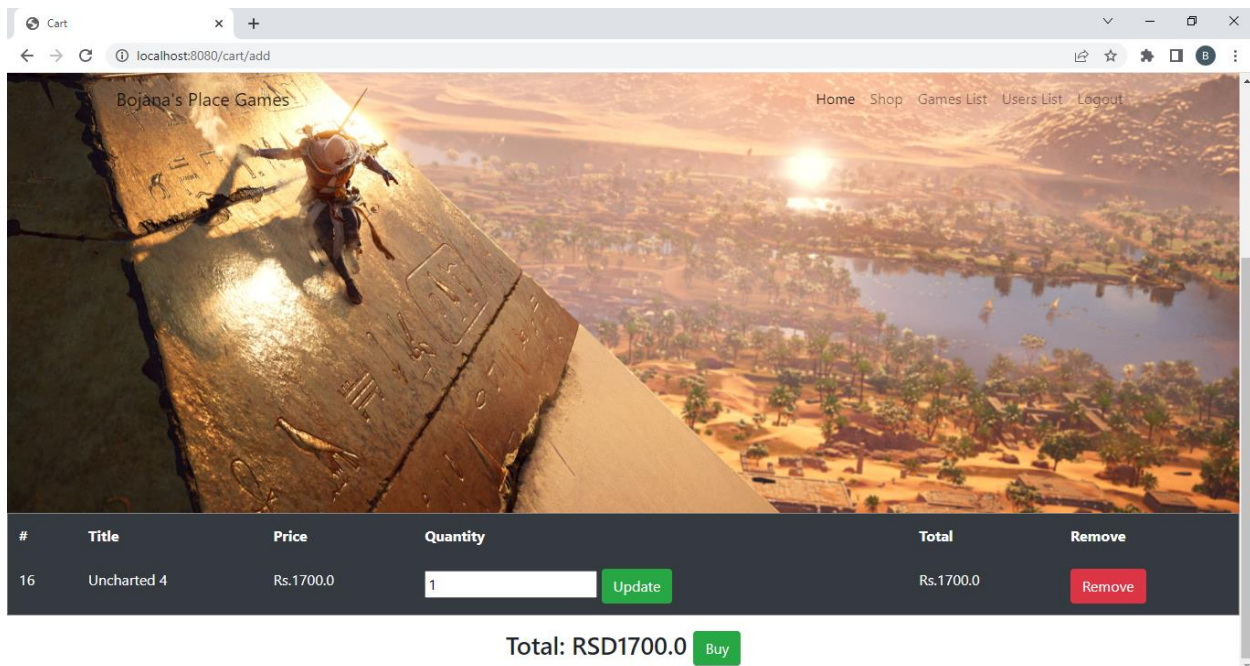
*Slika 10 – Pregled proizvoda na stranici za kupovinu



*Slika 11 – Pregled individualnog proizvoda klikom na njegov Id naglašen plavom bojom na prethodnoj slici

Korisnik klikom na Add to cart, dodaje proizvod u korpu što nas dalje vodi do korpa stranice.

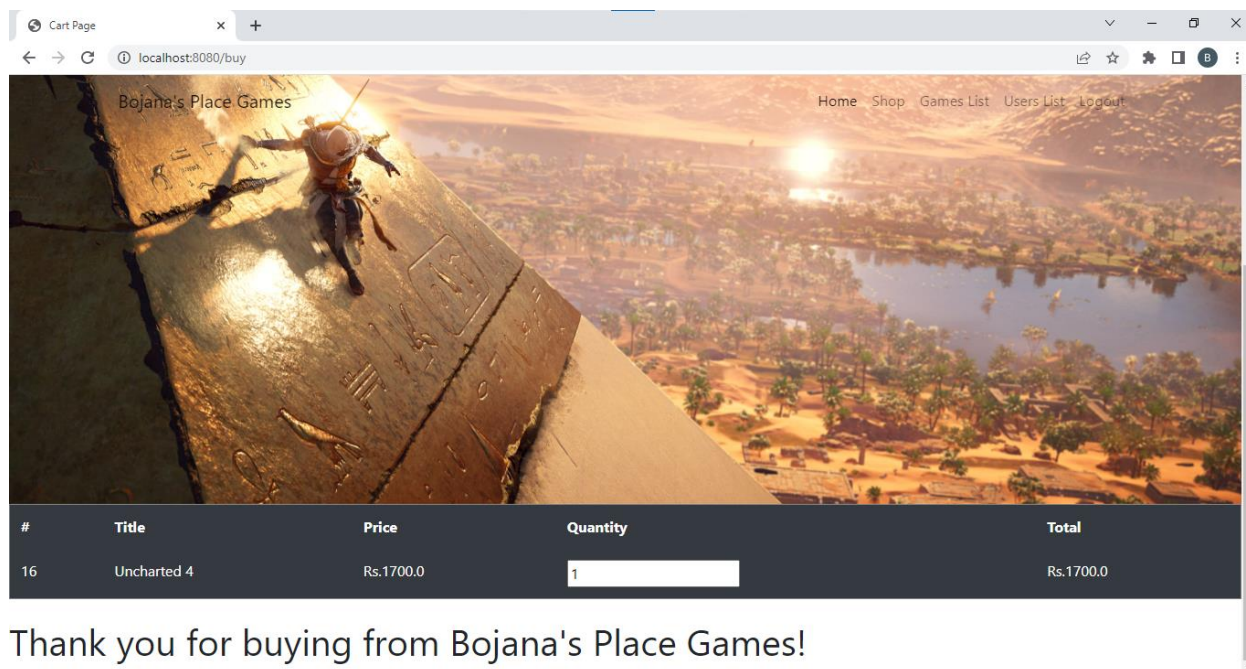
Korpa stranica



*Slika 12 – Korpa stranica

Na ovoj stranici možemo ukloniti proizvod iz korpe, promeniti količinu i kupiti sam proizvod.

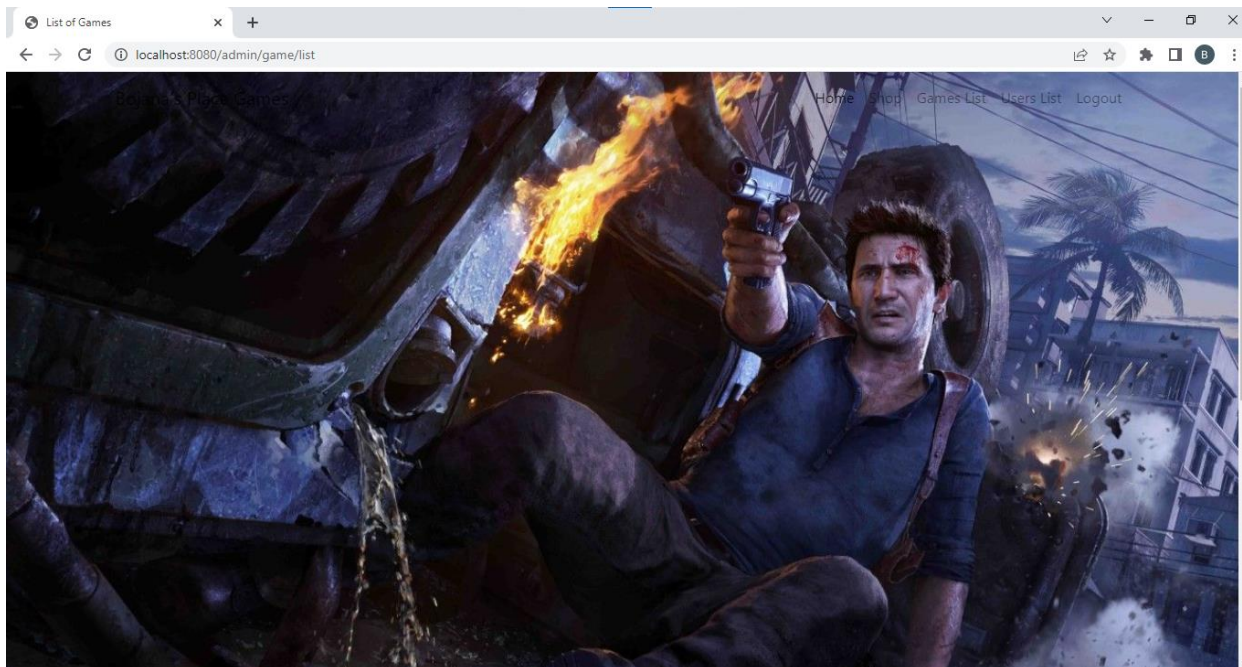
Realizovana kupovina stranica



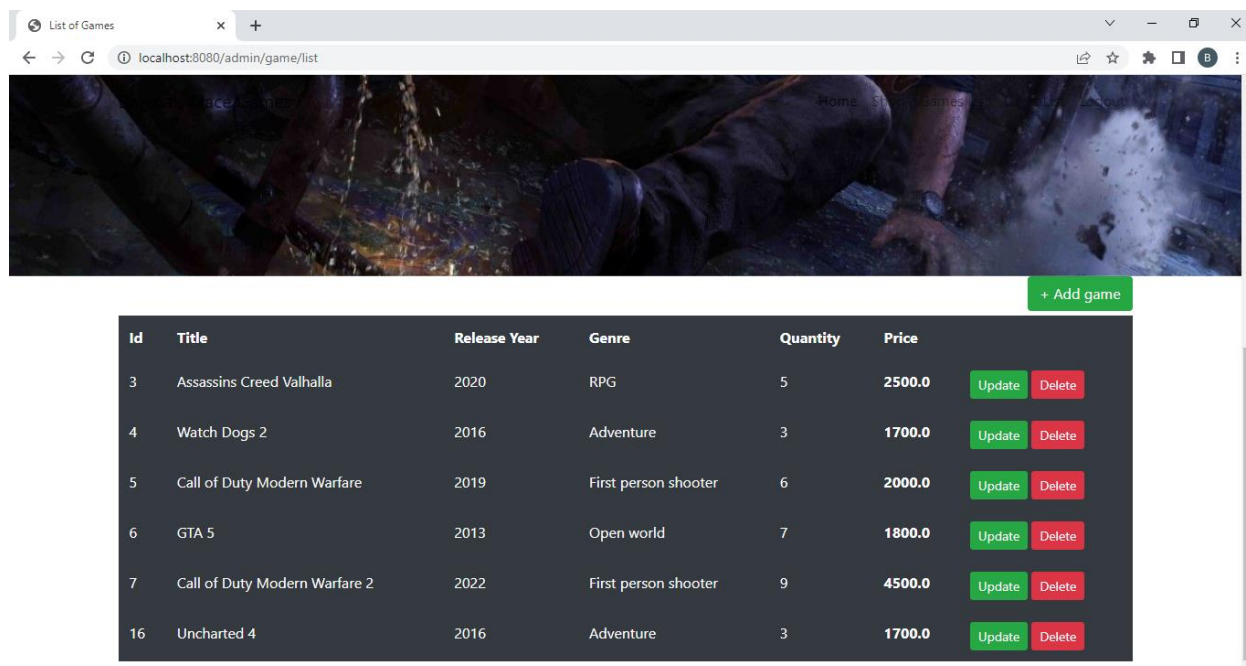
*Slika 13 – Uspešno realizovana kupovina stranica i poruka o istoj

Ukoliko običan korisnik pokuša da pristupi opcijama Games List i Users List sa navigacionog menija dobiće poruku da je to Admin only opcija. U nastavku, prijavljujemo se kao admin korisnik i prikazujemo njegove funkcionalnosti, izbegavajući kupovinu, jer je to već pokriveno na prethodnim slikama.

Games List stranica (admin korisnik)



*Slika 14 - Izgled i slika stranice za upravljanje podacima o igricama u ponudi, a kada scroll-ujemo dole dolazimo do tabele



*Slika 15 – Tabela za upravljanje podacima o igricama

Add/Edit Game

localhost:8080/admin/game/edit/3

Home Games Add Game Edit Game Games List Logout

Title:

Assassins Creed Valhalla

Release Year

2020

Genre

RPG

Quantity

5

Price

2500.0

Submit

*Slika 16 – Update stranica za igrice

Add/Edit Game

localhost:8080/admin/game/add

Home Games Add Game Edit Game Games List Logout

Title:

Title

Release Year

Release Year

Genre

Genre

Quantity

Quantity

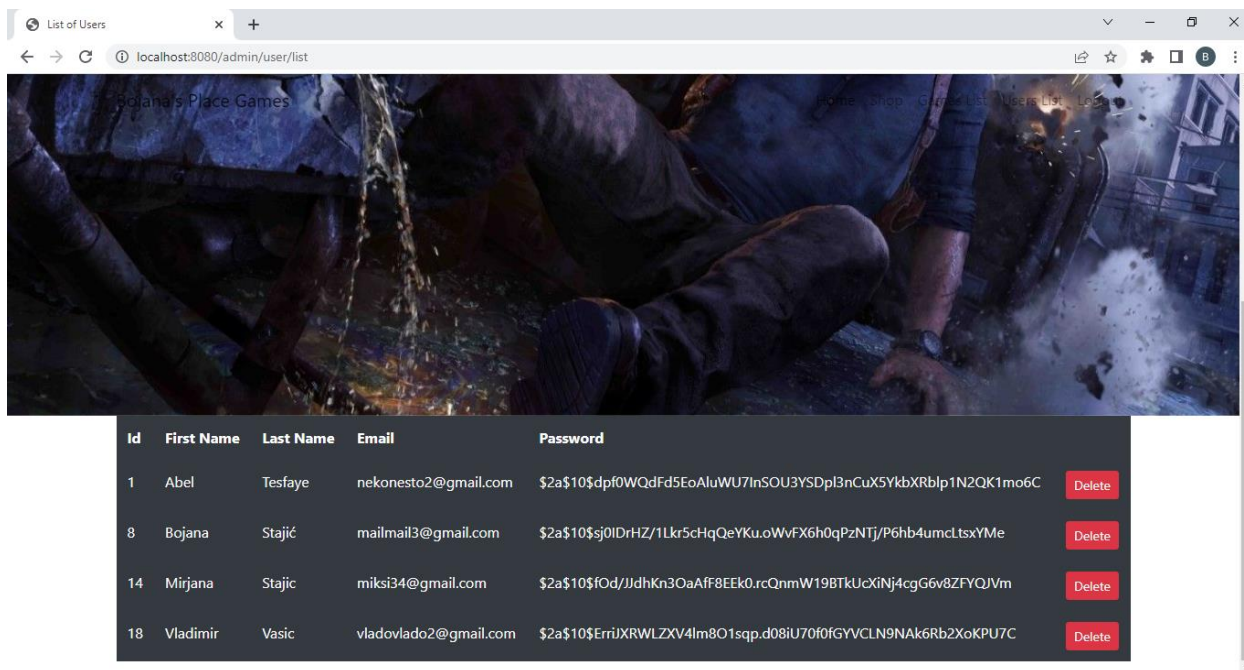
Price

Price

Submit

*Slika 17 – Add game/Dodaj igru stranica

Odabirom opcije Delete u Games List, igrica će biti obrisana i iz ponude i iz baze podataka, i tabela će se odmah refresh-ovati sa novim podacima.



*Slika 18 – Users List stranica

Admin na Users List stranici osim pregled korisnika sistema ima i mogućnost brisanja aktivnih korisnika kako iz sistema tako i iz baze podataka.

Klikom na dugme Logout, bez obzira da li smo admin ili ne, uspešno se odjavljujemo i preusmeravamo na početnu Login stranicu.

Testiranje

Tokom testiranja softvera, koristimo Junit biblioteku radi jediničnog testiranja aplikacije. Pored Junit-a biće korišćena i Mockito biblioteka koja pomaže u testiranju. U konkretnom slučaju testiranja ovog projekta, testirali smo korisničke/user delove aplikacije (UserRepository, UserService, User, ...).

```

@ExtendWith(MockitoExtension.class)
@SpringBootTest
public class UserServiceTest {

    3 usages
    @MockBean
    private UserRepository userRepository;

    1 usage
    @Autowired
    private UserService userService;

    @Test
    void saveUserTest() {
        User user = new User();
        Role role = new Role();
        BCryptPasswordEncoder passwordEncoder = new BCryptPasswordEncoder();
        role.setName("ROLE_USER");
        user.setPassword(passwordEncoder.encode( rawPassword: "test"));
        user.setEmail("test@test.com");

        when(userRepository.save(user)).thenReturn(user);
        System.out.println(user);
        assertEquals(user, userService.saveUser(user));
    }
}

```

*Slika 19 – kod test klase 1. deo

```

@Test
void getUserByEmailTest() {
    String email = "bojana32@yahoo.com";

    User user = new User();
    Role role = new Role();
    role.setName("ROLE_ADMIN");
    user.setPassword("sifra");
    user.setEmail(email);

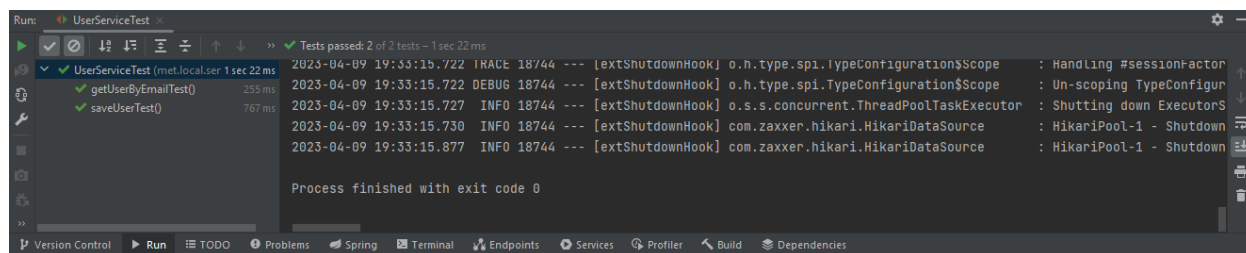
    when(userRepository.findByEmail(email)).thenReturn(user);

    User actualUser = userRepository.findByEmail(email);

    assertEquals(user, actualUser);
    assertEquals(email, actualUser.getEmail());
}
}

```

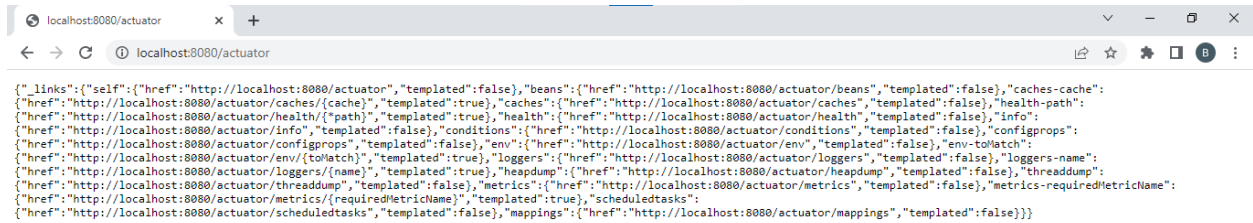
*Slika 20 – kod test klase 2. Deo



*Slika 21 – prikaz rezultata opisanih testova

Upotreba Aktuatora

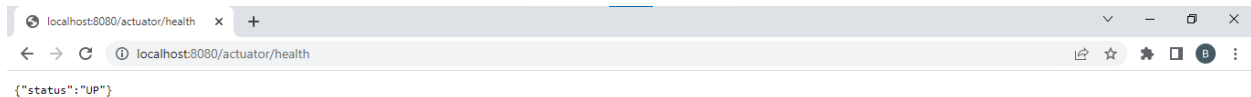
/actuator



A screenshot of a web browser window with the address bar showing 'localhost:8080/actuator'. The page content displays a JSON response from the Spring Actuator. The response includes links to various endpoints, beans, caches, health status, info, conditions, configprops, env, env-to-watch, loggers, heapdump, metrics, and scheduledtasks. The 'status' field is not explicitly shown in this snippet, but the 'health' field indicates the system is up.

```
{
  "links": {
    "self": {
      "href": "http://localhost:8080/actuator",
      "templated": false
    },
    "beans": {
      "href": "http://localhost:8080/actuator/beans",
      "templated": false
    },
    "caches-cache": {
      "href": "http://localhost:8080/actuator/caches/{cache}",
      "templated": true
    },
    "caches": {
      "href": "http://localhost:8080/actuator/caches",
      "templated": false
    },
    "health-path": {
      "href": "http://localhost:8080/actuator/health/{path}",
      "templated": true
    },
    "health": {
      "href": "http://localhost:8080/actuator/health",
      "templated": false
    },
    "info": {
      "href": "http://localhost:8080/actuator/info",
      "templated": false
    },
    "conditions": {
      "href": "http://localhost:8080/actuator/conditions",
      "templated": false
    },
    "configprops": {
      "href": "http://localhost:8080/actuator/configprops",
      "templated": false
    },
    "env": {
      "href": "http://localhost:8080/actuator/env",
      "templated": false
    },
    "env-to-watch": {
      "href": "http://localhost:8080/actuator/env/{toWatch}",
      "templated": true
    },
    "loggers": {
      "href": "http://localhost:8080/actuator/loggers",
      "templated": false
    },
    "loggers-name": {
      "href": "http://localhost:8080/actuator/loggers/{name}",
      "templated": true
    },
    "heapdump": {
      "href": "http://localhost:8080/actuator/heapdump",
      "templated": false
    },
    "threaddump": {
      "href": "http://localhost:8080/actuator/threaddump",
      "templated": false
    },
    "metrics": {
      "href": "http://localhost:8080/actuator/metrics",
      "templated": false
    },
    "metrics-requiredMetricName": {
      "href": "http://localhost:8080/actuator/metrics/{requiredMetricName}",
      "templated": true
    },
    "scheduledtasks": {
      "href": "http://localhost:8080/actuator/scheduledtasks",
      "templated": false
    },
    "mappings": {
      "href": "http://localhost:8080/actuator/mappings",
      "templated": false
    }
  }
}
```

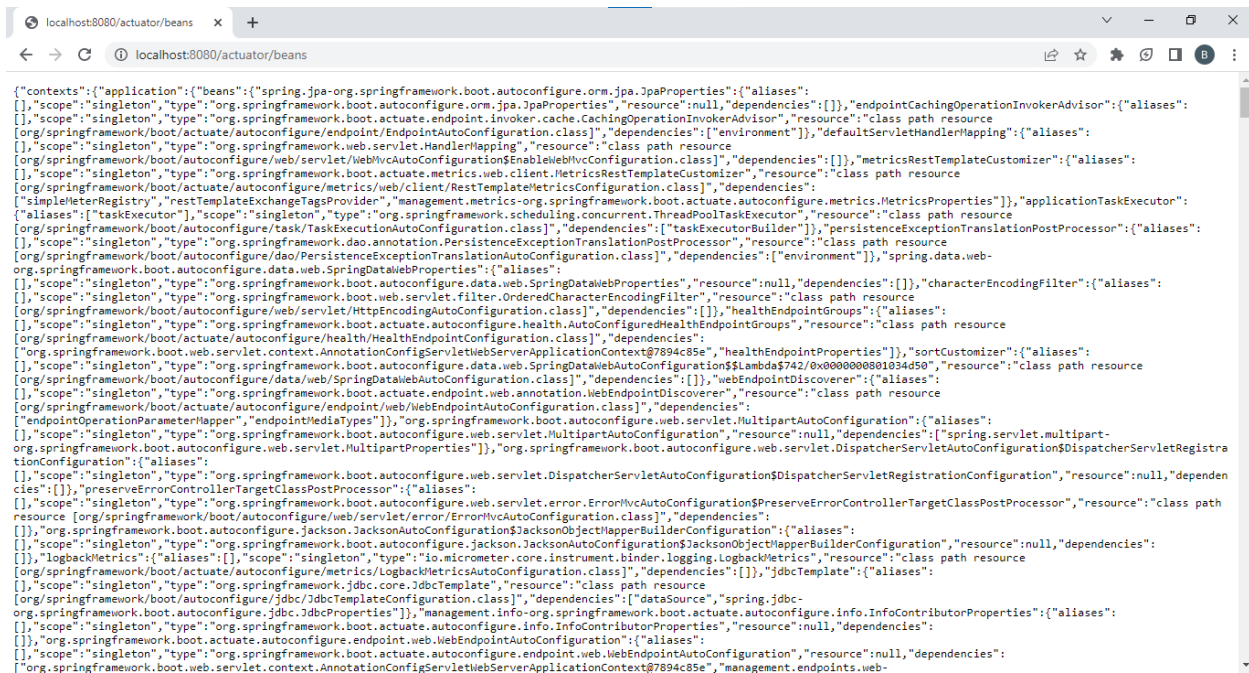
/health



A screenshot of a web browser window with the address bar showing 'localhost:8080/actuator/health'. The page content displays a simple JSON response indicating the system is up.

```
{
  "status": "UP"
}
```

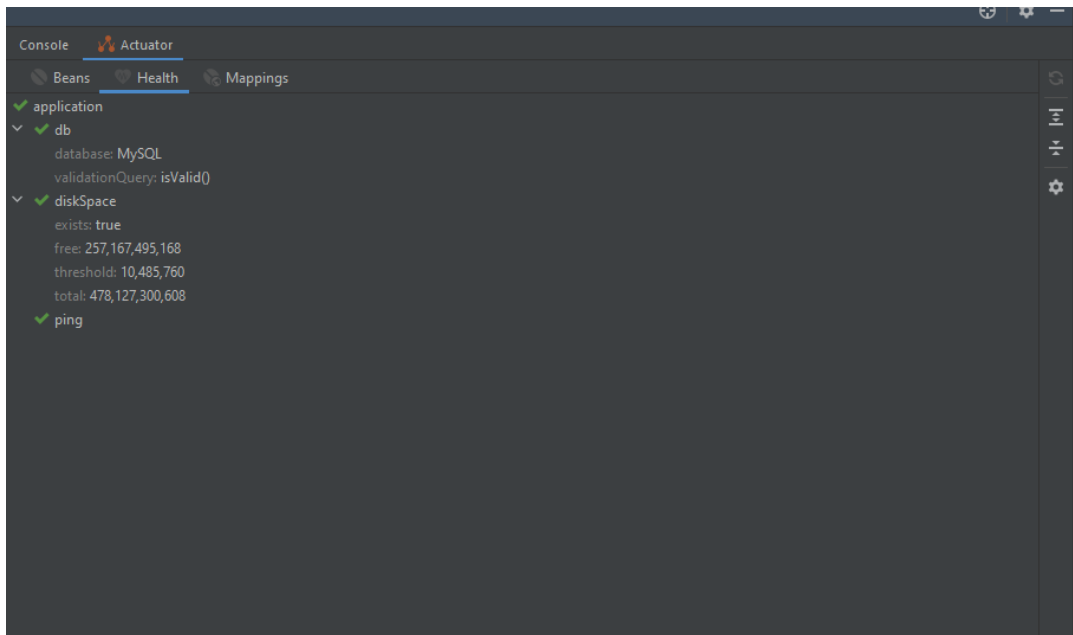
/beans



```
{
  "contexts": {
    "application": {
      "beans": {
        "spring.jpa-org.springframework.boot.autoconfigure.orm.jpa.JpaProperties": {
          "aliases": [
            {}
          ],
          "scope": "singleton",
          "type": "org.springframework.boot.autoconfigure.orm.jpa.JpaProperties",
          "resource": null,
          "dependencies": [
            {}
          ],
          "endpointCachingOperationInvokerAdvisor": {
            "aliases": [
            ],
            "scope": "singleton",
            "type": "org.springframework.boot.actuate.endpoint.invoker.cache.CachingOperationInvokerAdvisor",
            "resource": "class path resource",
            "dependencies": [
            ],
            "defaultServletHandlerMapping": {
              "aliases": [
              ],
              "scope": "singleton",
              "type": "org.springframework.web.servlet.HandlerMapping",
              "resource": "class path resource",
              "dependencies": [
                "environment"
              ],
              "metricsRestTemplateCustomizer": {
                "aliases": [
                ],
                "scope": "singleton",
                "type": "org.springframework.web.client.MetricsRestTemplateCustomizer",
                "resource": "class path resource",
                "dependencies": [
                ],
                "simpleMeterRegistry": {
                  "aliases": [
                  ],
                  "scope": "singleton",
                  "type": "org.springframework.boot.actuate.metrics.web.client.RestTemplateMetricsConfiguration.class",
                  "dependencies": [
                    "management.metrics-org.springframework.boot.actuate.autoconfigure.metrics.MetricsProperties"
                  ],
                  "applicationTaskExecutor": {
                    "aliases": [
                    ],
                    "scope": "singleton",
                    "type": "org.springframework.scheduling.concurrent.ThreadPoolTaskExecutor",
                    "resource": "class path resource",
                    "dependencies": [
                    ],
                    "taskExecutorBuilder": {
                      "aliases": [
                      ],
                      "scope": "singleton",
                      "type": "org.springframework.dao.annotation.PersistenceExceptionTranslationPostProcessor",
                      "resource": "class path resource",
                      "dependencies": [
                      ],
                      "spring.data.web-org.springframework.boot.autoconfigure.data.web.SpringDataWebProperties": {
                        "aliases": [
                        ],
                        "scope": "singleton",
                        "type": "org.springframework.boot.autoconfigure.data.web.SpringDataWebProperties",
                        "resource": null,
                        "dependencies": [
                        ],
                        "characterEncodingFilter": {
                          "aliases": [
                          ],
                          "scope": "singleton",
                          "type": "org.springframework.boot.web.servlet.filter.OrderedCharacterEncodingFilter",
                          "resource": "class path resource",
                          "dependencies": [
                          ],
                          "healthEndpointGroups": {
                            "aliases": [
                            ],
                            "scope": "singleton",
                            "type": "org.springframework.boot.actuate.autoconfigure.health.AutoConfiguredHealthEndpointGroups",
                            "resource": "class path resource",
                            "dependencies": [
                            ],
                            "org.springframework.boot.autoconfigure.health.HealthEndpointConfiguration.class": {
                              "aliases": [
                              ],
                              "scope": "singleton",
                              "type": "org.springframework.boot.autoconfigure.health.HealthEndpointConfiguration.class",
                              "resource": null,
                              "dependencies": [
                                "org.springframework.boot.web.servlet.context.AnnotationConfigServletWebServerApplicationContext$7894c85e",
                                "healthEndpointProperties"
                              ],
                              "sortCustomizer": {
                                "aliases": [
                                ],
                                "scope": "singleton",
                                "type": "org.springframework.boot.autoconfigure.data.web.SpringDataWebAutoConfiguration$$Lambda$742/0x0000000001034d50",
                                "resource": "class path resource",
                                "dependencies": [
                                ],
                                "webEndpointDiscoverer": {
                                  "aliases": [
                                  ],
                                  "scope": "singleton",
                                  "type": "org.springframework.boot.actuate.endpoint.web.annotation.WebEndpointDiscoverer",
                                  "resource": "class path resource",
                                  "dependencies": [
                                  ],
                                  "endpointOperationParameterMapper": {
                                    "aliases": [
                                    ],
                                    "scope": "singleton",
                                    "type": "org.springframework.boot.actuate.endpoint.web.servlet.MultipartAutoConfiguration",
                                    "resource": null,
                                    "dependencies": [
                                      "spring.servlet.multipart-org.springframework.boot.autoconfigure.web.servlet.MultipartProperties"
                                    ],
                                    "org.springframework.boot.autoconfigure.web.servlet.DispatcherServletAutoConfiguration$DispatcherServletRegistrationConfiguration": {
                                      "aliases": [
                                      ],
                                      "scope": "singleton",
                                      "type": "org.springframework.boot.autoconfigure.web.servlet.DispatcherServletAutoConfiguration$DispatcherServletRegistrationConfiguration",
                                      "resource": null,
                                      "dependencies": [
                                      ],
                                      "preserveErrorControllerTargetClassPostProcessor": {
                                        "aliases": [
                                        ],
                                        "scope": "singleton",
                                        "type": "org.springframework.boot.autoconfigure.web.servlet.error.ErrorMvcAutoConfiguration$PreserveErrorControllerTargetClassPostProcessor",
                                        "resource": "class path resource",
                                        "dependencies": [
                                        ],
                                        "org.springframework.boot.autoconfigure.jackson.JacksonAutoConfiguration$JacksonObjectMapperBuilderConfiguration": {
                                          "aliases": [
                                          ],
                                          "scope": "singleton",
                                          "type": "org.springframework.boot.autoconfigure.jackson.JacksonAutoConfiguration$JacksonObjectMapperBuilderConfiguration",
                                          "resource": null,
                                          "dependencies": [
                                          ],
                                          "logbackMetrics": {
                                            "aliases": [
                                            ],
                                            "scope": "singleton",
                                            "type": "io.micrometer.core.instrument.binder.logging.LogbackMetrics",
                                            "resource": "class path resource",
                                            "dependencies": [
                                            ],
                                            "jdbcTemplate": {
                                              "aliases": [
                                              ],
                                              "scope": "singleton",
                                              "type": "org.springframework.jdbc.core.JdbcTemplate",
                                              "resource": "class path resource",
                                              "dependencies": [
                                                "dataSource",
                                                "spring.jdbc-org.springframework.boot.autoconfigure.jdbc.JdbcProperties"
                                              ],
                                              "management.info-org.springframework.boot.actuate.autoconfigure.info.InfoContributorProperties": {
                                                "aliases": [
                                                ],
                                                "scope": "singleton",
                                                "type": "org.springframework.boot.actuate.autoconfigure.info.InfoContributorProperties",
                                                "resource": null,
                                                "dependencies": [
                                                ],
                                                "org.springframework.boot.actuate.autoconfigure.endpoint.web.WebEndpointAutoConfiguration": {
                                                  "aliases": [
                                                  ],
                                                  "scope": "singleton",
                                                  "type": "org.springframework.boot.actuate.autoconfigure.endpoint.web.WebEndpointAutoConfiguration",
                                                  "resource": null,
                                                  "dependencies": [
                                                  ],
                                                  "org.springframework.boot.web.servlet.context.AnnotationConfigServletWebServerApplicationContext$7894c85e",
                                                  "management.endpoints.web-"
                                                ]
                                              ]
                                            }
                                          ]
                                        }
                                      }
                                    }
                                  }
                                }
                              }
                            }
                          }
                        }
                      }
                    }
                  }
                }
              }
            }
          ]
        }
      }
    }
  }
}
```

Aktuator je Spring Boot mehanizam koji nudi alate spremne za produkciju aplikacije. Upotrebom određenih komandi u kontekstnoj putanji dobijamo različite unutrašnje detalje vezane za sam projekat. Na slikama iznad, prikazana je default actuator stranica, zatim i upotreba /health putanje kao i /beans putanje za izveštaj o rasporedu i povezanosti zrna na ovom projektu. Postoje još mnoge korisne funkcije koje se mogu pozvati nakon što smo dodali odgovarajuću zavisnost kao što jesmo.

Prikaz aktuatora unutar razvojnog okruženja

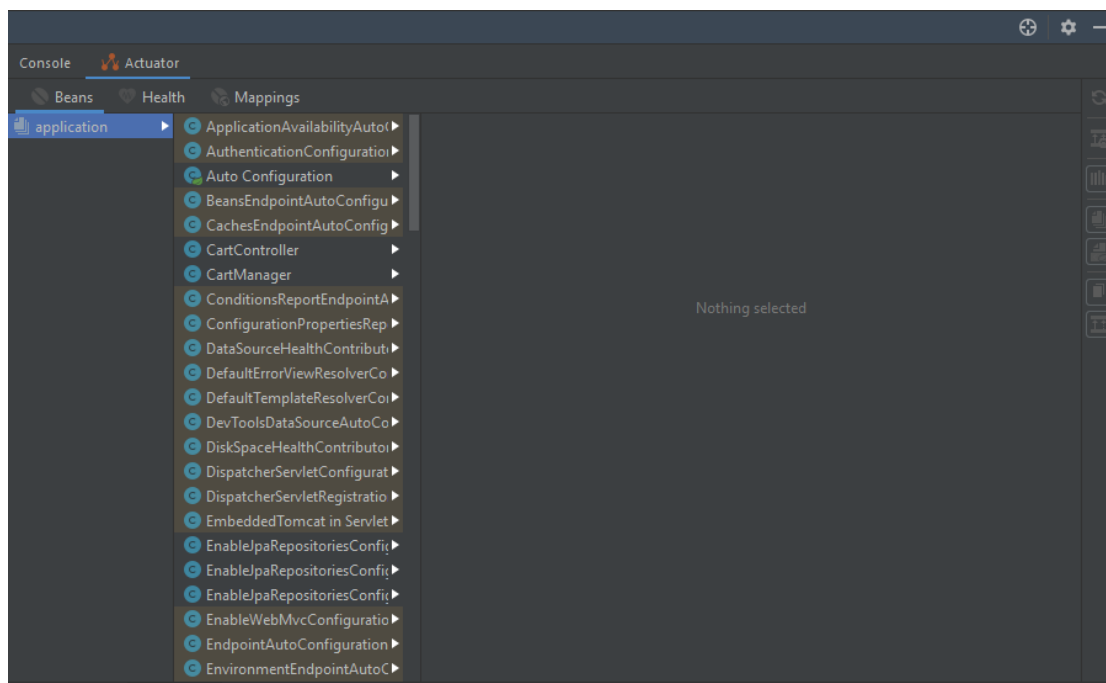


*Health

The screenshot shows the Spring Actuator Mappings endpoint. The 'Mappings' tab is selected, displaying a table of all available endpoints and their corresponding methods.

Path	Method
/ [GET]	MainController#home
/**	
/actuator [GET]	WebMvcLinksHandler#links
/actuator/beans [GET]	OperationHandler#handle
/actuator/caches [GET]	OperationHandler#handle
/actuator/caches [DELETE]	OperationHandler#handle
/actuator/caches/{cache} [GET]	OperationHandler#handle
/actuator/caches/{cache} [DELETE]	OperationHandler#handle
/actuator/conditions [GET]	OperationHandler#handle
/actuator/configprops [GET]	OperationHandler#handle
/actuator/env [GET]	OperationHandler#handle
/actuator/env/{toMatch} [GET]	OperationHandler#handle
/actuator/health [GET]	OperationHandler#handle
/actuator/health/** [GET]	OperationHandler#handle
/actuator/heapdump [GET]	OperationHandler#handle
/actuator/info [GET]	OperationHandler#handle
/actuator/loggers [GET]	OperationHandler#handle
/actuator/loggers/{name} [POST]	OperationHandler#handle

*Mappings



*Beans

Zaključak

U zaključku, kreiran je jedan projekat koji je zaokružio priču o osnovama Spring-a a i nadogradio teme iz prethodnog gradiva. Rezultat jeste jedan kvalitetna i kompletna aplikacija koja sadrži kako front-end tako i back-end dalje ispunjavajući korisničke zahteve i zahteve definisane apstraktnom projektnog zadatka. Aplikacija bi se u budućnosti mogla unaprediti bolje organizovanom i detaljnijom bazom podataka.

Source code

GitHub link ka projektu: <https://github.com/little-software-engineer/IT355-PZ-BojanaStajic4596>

Literatura

1. LAMS Nastavni materijali predmeta IT355- Veb Sistemi 2
2. Gary Mak, Josh Long, and Daniel Rubio, Spring Recipes Third Edition, Apress
3. Spring Framework Reference Documentation – <https://docs.spring.io/spring-framework/docs/current/spring-framework-reference/>