Vladyslav Holubenko
ID: 102949

Objektovo-orientované programovanie

Internetový obchod hudobných nástrojov *RockShop*

1. Zámer projektu

V tomto projekte sa bude realizovať internetový obchod z predaju viacerých hudobných nástrojov(gitara, bubon, syntetizér).

Internetový obchod bude mať hlavnú stránku z celým zoznamom tovarov, ktoré je možne usporiadať podľa rôznych kritérií na záujem používateľa. Tým pádom klient si môže objednať akýkoľvek tovar alebo, proste, dať ho do košíka (keď on ho zaujíma). Navyše každý zákazník bude mať vlastný súkromný kabinet. V ňom on môže riadiť všetkými svojimi zákazkami (napr. zmazať nepotrebný tovar). Kedy používateľ si vyberie niekoľko tovarov, tak na konce bude vyrátaná celková suma cen tých tovarov. Pri tvorbe konečnej objednávky každí zákazník bude musieť uviesť personál info (svoju adresu). Celým systémom riadi administrátor. On má možnosť pridávať tovar k databáze, alebo vylúčiť ho. Tak tiež administrátor ma prístup k informácii o zákazníkoch.

Po ukončení transakcie servis výtvory účtenku s celkovou sumou nákupu a personálnym číslom operácii.

2. Štruktúra systému

Hierarchia Product:

Na vrchole hierarchie rozhranie IProduct v ktorom zadefinovane hlavne metódy. Rozhranie IProduct realizuje abstraktný metód Product. Od Product sú zdedene Drum, Guitar a Synthesizer, ktoré dedia rozhrania IGuitar, IDrum, ISynhesizer a ktoré vlastne sú tabuľky databázy, kde zapísane tovary, predstavené, ako objekty(Obrázok 2.1).

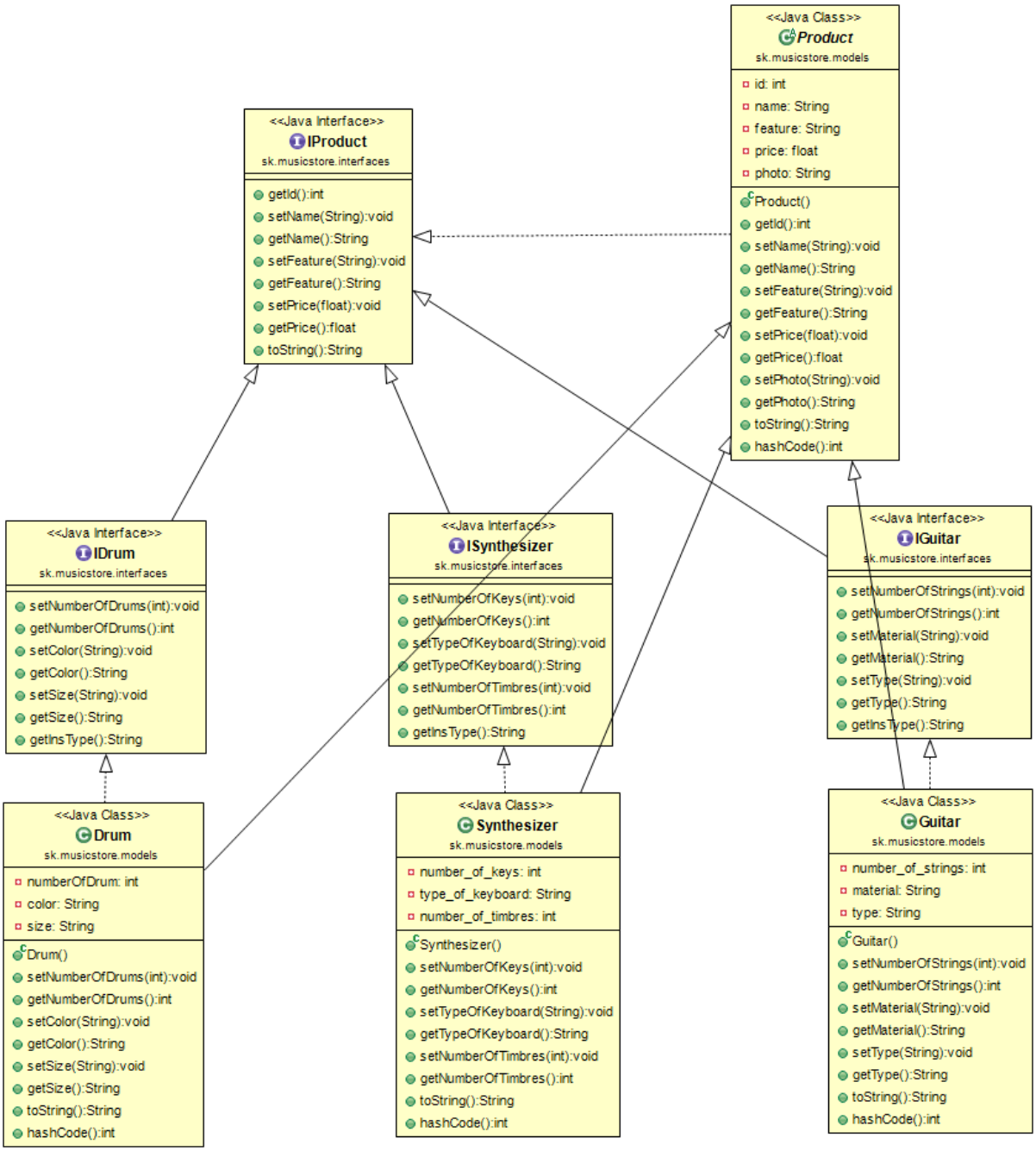
Hierarchie Order:

Na vrchole hierarchie rozhranie IOrder v ktorom zadefinovane hlavne metódy. Rozhranie IOrder realizuje abstraktný metód Order. Od Order sú zdedene OrderDrum, OrderGuitar a OrderSynthesizer, ktorý vlastne sú tabuľky databázy, kde sú uložene kúpene tovary, predstavené, ako objekty(Obrázok 2.2).

Hierarchie OrderUser:

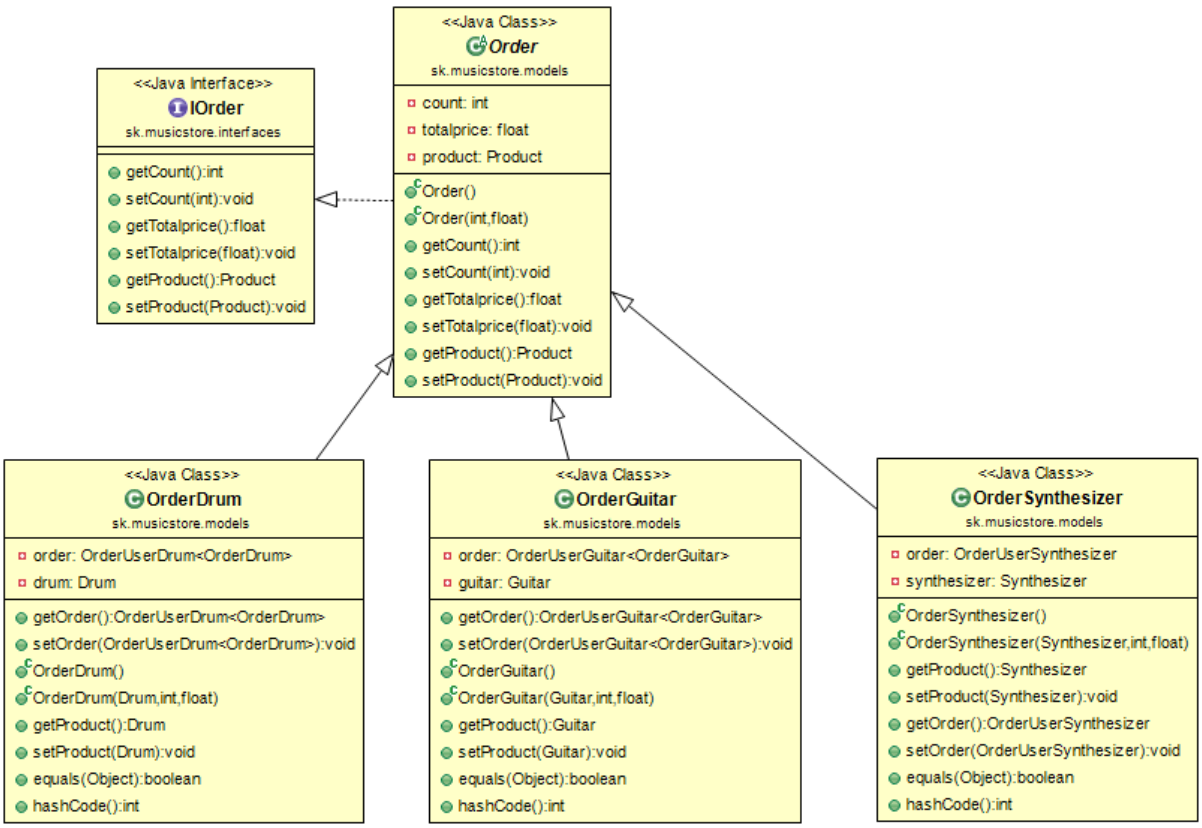
Na vrchole hierarchie rozhranie IOrderUser v ktorom zadefinovane hlavne metódy. Rozhranie IOrderUSer realizuje abstraktný metód OrderUser. Od OrderUser sú zdedene OrderUserDrum, OrderUserGuitar a OrderUserSynthesizer, ktorý vlastne sú tabuľky databázy, kde sú uložený zákazníci tovary, predstavené, ako objekty(Obrázok 2.3).

Vladyslav Holubenko
ID: 102949

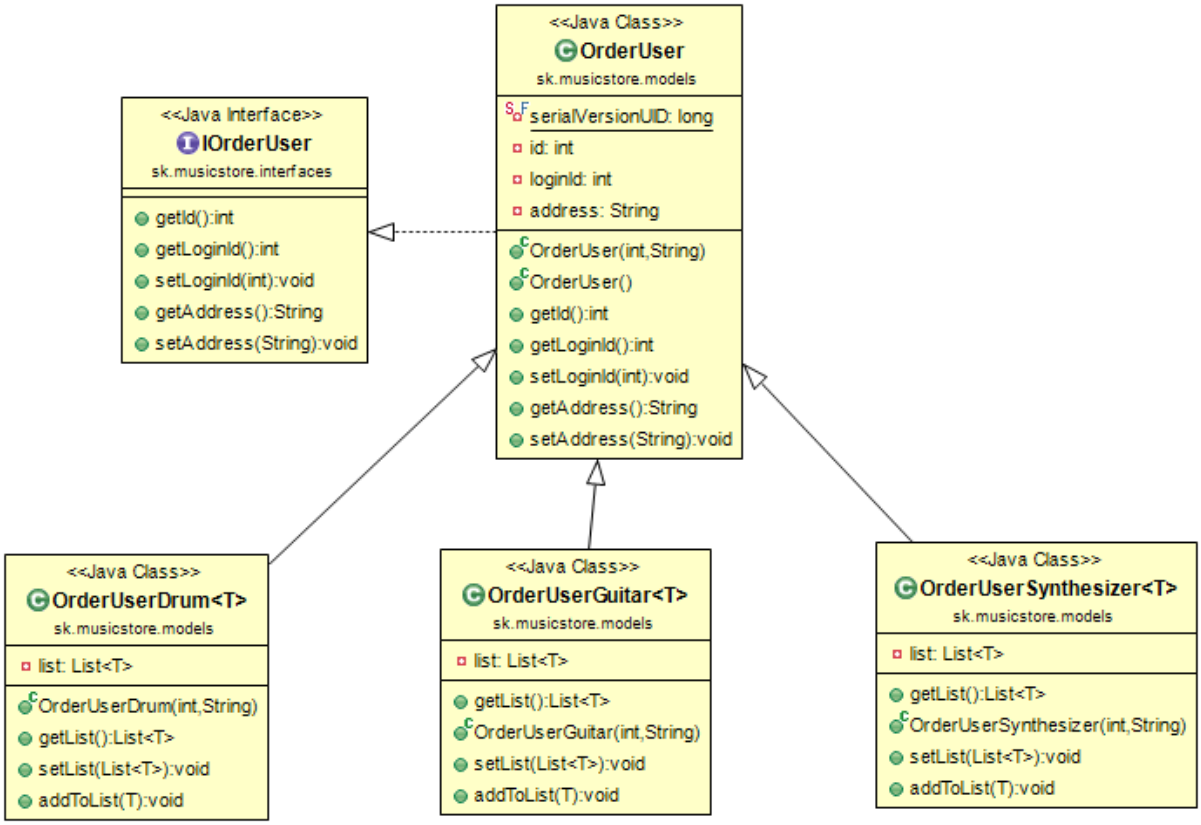


Obrázok 2.1 Hierarchia Product

Vladyslav Holubenko
ID: 102949



Obrázok 2.2 Hierarchia Order



Obrázok 2.3 Hierarchia OrderUser

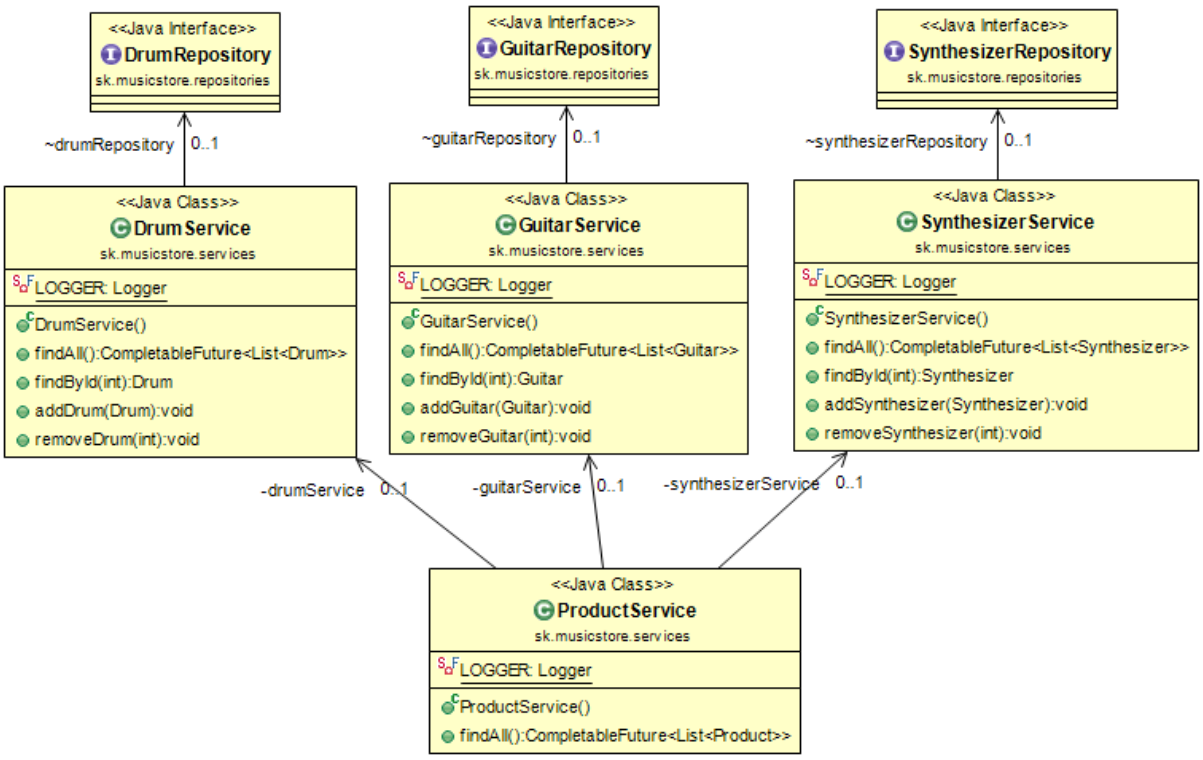
Vladyslav Holubenko
ID: 102949

Hierarchia ProductService:

Rozhrania DrumRepository, GuitarRepository, SynthesizerRepository dedia rozhranie JpaRepository a sú použitý pre prístupu do databázy. JpaRepository je rozhranie Spring Data Framework, ktoré poskytuje sadu štandardných metód JPA pre prácu s databázou. Na základe tohto rozhrania Spring Data poskytne implementovane metódy.

Triedy DrumService, GuitarService, SynthesizerService sú použitý na zaistenie logiky prace s údajmi. Oni majú agregované rozhrania DrumRepository, GuitarRepository a SynthesizeRepository.

Trieda ProductService má agregované triedy DrumService, GuitarService, SynthesizerService a určená, aby vrátiť zoznam tovarov všetkých typov(Obrázok 2.4).



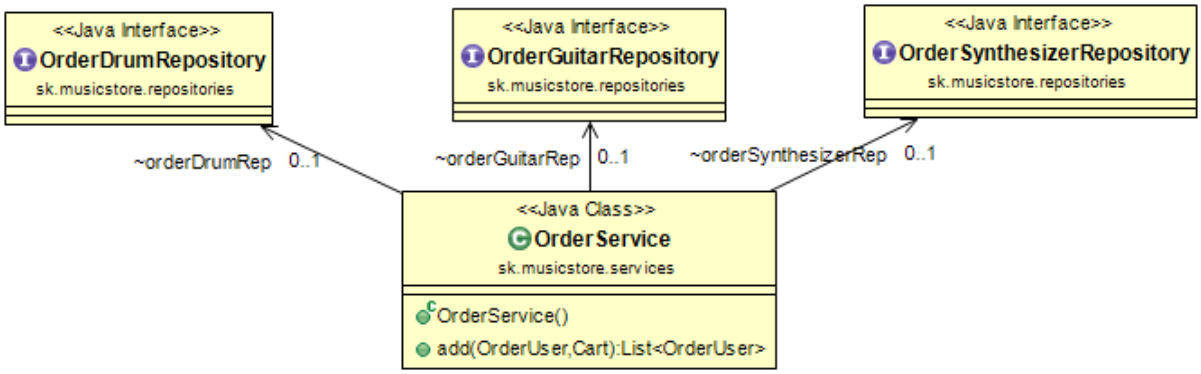
Obrázok 2.4 Hierarchia ProductServices

Hierarchia OrderService:

Rozhrania OrderDrumRepository, OrderGuitarRepository, OrderSynthesizerRepository dedia rozhranie JpaRepository a sú použitý pre prístupu do databázy.

Trieda ProductService má agregované rozhrania OrderDrumRepository, OrderGuitarRepository, OrderSynthesizerRepository a určená, aby dodať novu objednávku do databázy (Obrázok 2.5).

Vladyslav Holubenko
ID: 102949

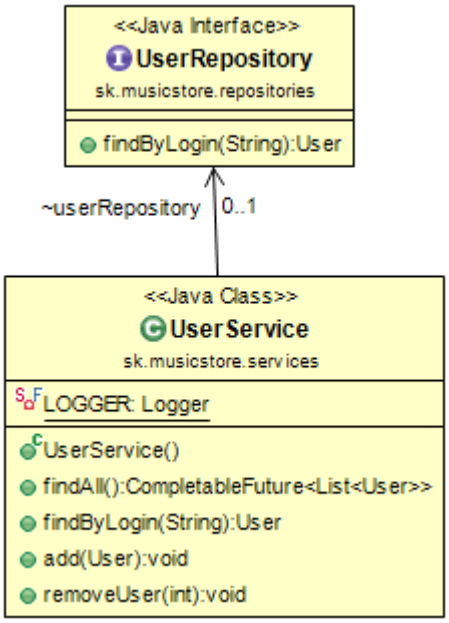


Obrázok 2.5 Hierarchia OrderService

Hierarchia UserService:

Rozhranie UserRepository dedi rozhranie JpaRepository a sú použitý pre prístupu do databázy.

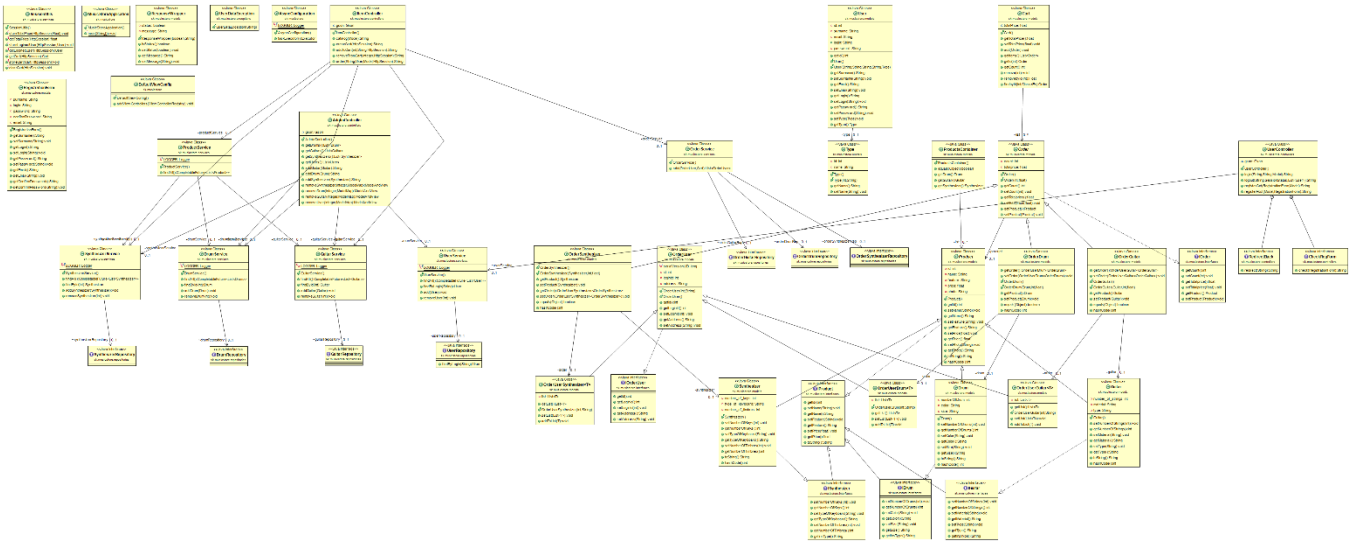
Trieda UserService má agregované rozhranie UserRepository a určená, aby dodať nového zákazníka do databázy (Obrázok 2.6).



Obrázok 2.6 Hierarchia UserService

Všeobecný diagram závislosti:

Vladyslav Holubenko
ID: 102949



3. Splnene kritéria

Projekt bol vytvorený s pomocou Spring Boot + Hibernate + MySQL + Bootstrap a použitý MVC pattern.

Hlavne kriteria:

Miera splnenia projektu je plna. Boli uplatnený objektovo-orientovaný mechanizmy v programe, a pravé dedenie, polymorfizmus, zapuzdrenie a agregácia v hierarchiách, čo boli uvedený vyššie. Kód má dobrú organizáciu a kvalitnú dokumentáciu.

Ďalšie kritéria:

- Vlastná výnimka

Bola vytvorená vlastná výnimka *UserDataException.*

```
public class UserDataException extends Exception {
    public UserDataException(String message){
        super(message);
    }
}
```

Ona vyhadzuje v triede UserService, ak zákazník nič neuviedol do polia Login.

```
public User findByLogin(String login) throws UserDataException{
            if(login.equals("")) throw new UserDataException("Login is empty");
        return userRepository.findByLogin(login);
    }
```

Ona je ošetrovaná v triede UserController

```
try {
            user=userService.findByLogin(login);
    }catch(UserDataException ex) {
return gson.toJson(new ResponseWrapper(false,ex.getMessage()));
    }
```

- Grafické používateľské rozhranie

Vladyslav Holubenko
ID: 102949

Pre vytvorenie vlastného GUI použil som HTML, CSS a framework Bootstrap. Pre generovania HTML rozmerania tabúľ bol použitý JQuery.

Súbory GUI nachádzajú sa v priečinku \src\main\webapp\.

- Multithreading

Multithreading použil som prostredníctvom API, a vlastne triedy CompletableFuture. CompletableFuture - trieda pre asynchrónnu prevádzku, ktorá umožňuje kombinovať kroky spracovania a spájať ich do reťazca.

Anotácia @EnableAsync umožňuje Spring schopnosti spúšťať metódy @Async v oblasti vlákien pozadia. Bean taskExecutor pomáha prispôsobiť executor vlákien, ako je napríklad konfigurácia počtu vlákien pre aplikáciu, limitná veľkosť frontu atď. Spring bude konkrétne hľadať túto Bean pri spustení servera.

```
@EnableAsync
public class AsyncConfiguration {
    private static final Logger LOGGER = LoggerFactory.getLogger(AsyncConfi
guration.class);
    @Bean (name = "taskExecutor")
    public Executor taskExecutor() {
        LOGGER.debug("Creating Async Task Executor");
        final ThreadPoolTaskExecutor executor = new ThreadPoolTaskExecutor(
);
        executor.setCorePoolSize(2);
        executor.setMaxPoolSize(2);
        executor.setQueueCapacity(100);
        executor.setThreadNamePrefix("CarThread-");
        executor.initialize();
        return executor;
    }
}
```

Anotácia @Async ukáže, aký metód bude fungovať asynchronicky. Asynchronicky funguje metód findAll() v ProductService, DrumService, GuitarService a SynthesizerService.

```
@Async
	public CompletableFuture<List<Drum>> findAll(){
		CompletableFuture<List<Drum>>
list=CompletableFuture.completedFuture(drumRepository.findAll());
		LOGGER.info("Request to get a list of drums");
		return list;
	}
```

- Generickosť

Bola realizovaná generickosť v triedach OrderUserDrum, OrderUserGuitar a OrderUserSynthesizer.

```
public class OrderUserGuitar<T> extends OrderUser
```

- RTTI

Vladyslav Holubenko
ID: 102949

RTTI bolo použite v triede OrderService, aby získať do ktorého typu patri produkt v Cart.

```
if(or instanceof OrderDrum ) {
	((OrderDrum)or).setOrder(orderDrum);
	orderDrum.addToList((OrderDrum)or);
}
else if(or instanceof OrderGuitar ) {
	((OrderGuitar)or).setOrder(orderGuitar);
	orderGuitar.addToList((OrderGuitar)or);
}
else if(or instanceof OrderSynthesizer){
	((OrderSynthesizer)or).setOrder(orderSynthesizer);
	orderSynthesizer.addToList((OrderSynthesizer)or);
}
```

- Vhniezdená trieda a rozhranie

Vhniezdený rozhrania sú v triede UserController.

```
interface RedirectBack{
    String redirect(String ref);
}

interface CheckRegForm{
    String check(RegistrationForm form);
}
```

- Lambda výrazy

Lambdá výrazy sú v triede UserController. V metóde logout(...) lambdá výraz na to, aby riešiť kam bude presmerovaný zákazník po východu z účtu.

```
RedirectBack redirectBack = (String ref)-> {
              if (ref != null && ref.contains("/account")) {
                   return "redirect:/";
              } else {
                   return "redirect:"+ref;
              }
       };
```

V metóde registerPost, aby overiť či uvedene všetci polia registrovania.

```
CheckRegForm checkRegForm=(RegistrationForm form)->{
             if(regForm.getSurname()=="") return "Enter surname";
             else if(regForm.getEmail()=="") return "Enter email";
             else if(regForm.getLogin()=="") return "Enter login";
             else if(regForm.getPassword()=="") return "Enter password";
             else
if(regForm.getPassword().equals(regForm.getConfirmPassword())) return "The
passwords are different!";
             else return null;
       };
```

- Default method implementation

Použil som default method implementation v rozhraniach IGuitar, IDrum a ISynthesizer.
A oni tam na to, aby bola možnosť spoznať na stránke, akého typu je objekt, kedy odošlem zoznam tovarov zo serveru.

```
public default String getInsType() {
	return "drum";
}

public default String getInsType() {
	return "guitar";
}

public default String getInsType() {
	return "synthesizer";
}
```

- Serializácia

Serializácia bola použitá v triedach Drum, Guitar, Synthesizer, User a Cart na to, aby mohol by som odoslať zoznam objektov tovarov, usera a kôš na stránku zo serveru.

```
public class User implements Serializable

public class Drum extends Product implements IDrum, Serializable

public class  Guitar extends Product implements IGuitar, Serializable

public class Synthesizer extends Product implements ISynthesizer, Serializable

	public class Cart implements Serializable
```

4. Hlavne verzii programu na GitHube

1. commit d858356d6454d2094a97682f882b157122580668
   Date: Sat Apr 4 19:23:38 2020 +0200
   Funguje zobrazenie zoznamu tovarov. Pridaný controllers, repositories a services pre tovarov.
2. commit fb9a905f23fc6da8c1f23fb15f31dab956616be8
   Date: Tue Apr 7 19:20:49 2020 +0200
   Fungujú funkcii autorizácii a registrácii.
3. commit 462723904779a9653e6c20dc44774b991baa3436
   Date: Mon May 18 17:35:36 2020 +0200
   Pridal som funkcii objednávky.
4. commit 052888685e0e381d2814c06cd27d0a79f0ee92fa
   Date: Tue May 19 00:05:14 2020 +0200
   Pridal som splnene kritéria(lambdá výraz, výnimka, generickosť, default method implementation, vhniezdený rozharania) do programu.
5. commit 9af00a4e882f16d77e989cc03cae1e963d2eb8cf
   Date: Sun May 24 21:11:23 2020 +0200
   Pridal som funkciu dodávania tovaru a zmazania tovaru.