Fakultet elektrotehnike i računarstva

**Objektno oblikovanje**

**SEMINAR**

**LEGO Manager**

**Sustav za upravljanje inventarom LEGO kocki**

Helena Tamburić

Brigita Vrbanec

Aleksandar Vezmar

Nikola Miličić

Luka Čubelić

Zagreb, 1. veljače 2017.

# **Design story**

Lego Manager je sustav koji omogućuje ljubiteljima Lego kocki lakše upravljanje vlastitim inventarom. Korisnici aplikacije skupljaju Lego kocke i setove. Set je skup Lego kocaka koji pripada određenoj temi i kategoriji, te je namjenjen slaganju određene kreacije. Osnovno svojstvo Lego kocki je mogućnost njihovog korištenja i kombiniranja na različite načine. Primarna namjena Lego Manager sustava je upravo olakšavanje prenamjene Lego kocki u korisnikovom vlasništvu.

Temeljna funkcionalnost sustava je predlaganje setova koje je moguće složiti od kocki koje korisnik ima na raspolaganju. Pri tome korisnik ima mogućnost izbora želi li za slaganje novog seta koristiti sve kocke koje posjeduje, ili samo one koje trenutno nisu iskorištene za slaganje nekog drugog seta. Korisnik također može ograničiti predlaganje samo na setove iz određene teme, odnosno kategorije, kao i na setove objavljene u zadanom vremenskom rasponu. Za sve setove predložene od strane sustava, korisniku su na raspolaganju i potrebne upute za slaganje. Kupnjom većeg broja setova veća je mogućnost da će korisnik od istih kocki moći sastaviti i neki drugi set koji ne posjeduje.

Svaku kupnju Lego kocki moguće je evidentirati u aplikaciji. Kupnjom seta automatski se evidentira posjedovanje svih kocki sadržanih u tom setu. Moguće je istovremeno evidentirati kupnju više primjeraka nekog seta ili kocke. Korisnici kupuju više primjeraka istog seta primarno zbog rijetkih dijelova sadržanih u tom setu kojima će možda moći složiti neke druge setove.

# Baza podataka

Baza Lego proizvoda se u stvarnosti mijenja samo nekoliko puta godišnje, stoga je moguće podržati i pregled svih Lego proizvoda koji su do tada izašli. Korisnik može po određenim kriterijima (tema, podtema, godina izdavanja) pretraživati Lego proizvode, kako setove tako i zasebne kocke. Svaki korisnik ima i vlastite liste želja te favorita na koje može dodavati Lego proizvode.

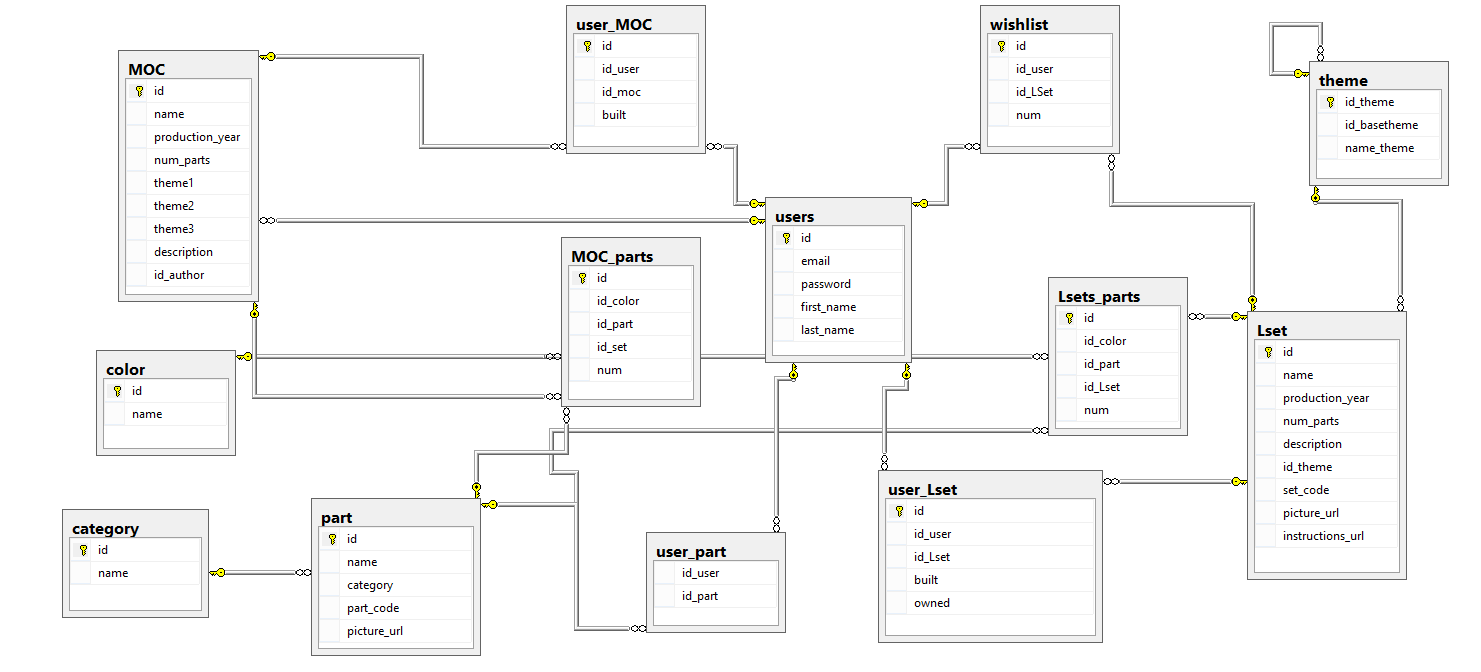
Korisniku je omogućen i unos samostalno dizajniranih setova (MOC – My Own Creation), pri čemu je potrebno navesti sve potrebne kocke. Isto tako, korisnik može pregledavati neslužbene setove, definirane od strane drugih korisnika sustava.

Sustav bi trebao nuditi i svojevrsnu zajednicu za ljubitelje Lego kocki. Pravi entuzijasti se mogu pohvaliti svojim kolekcijama Lego kocki, kao i vlastim kreacijama putem svog korisničkog profila. S obzirom da je glavna ideja olakšano vođenje evidencije, te predlaganje novih setova za slaganje, sustav ne nudi uslugu kupnje i prodaje Lego kocki.

Osnova našeg sustava je relacijska baza podataka Lego setova i kocki koja je popunjena podacima koji su javno dostupni na više mrežnih mjesta te se s istih može preuzeti (npr. www.rebrickable.com). U bazi su zabilježeni podaci o gotovo svim Lego setovima koji su ikad izdani, te o gotovo svim kockicama koje su ikad napravljene. Da bi pomoćnik za slaganje odredio koje sve setove korisnik može složiti od odabranih kocki, on prima odabrani podskup kocki te ga uspoređuje sa potrebnim kockama za slaganje drugih setova. U zavisnosti od odabira korisnika, pomoćnik ili uzima ili ne uzima u obzir boje samih kockica kod uspoređivanja. Čim otkrije da kocka koja je potrebna za slaganje nekog seta ne postoji u odabranom podskupu taj se set eliminira iz razmatranja. Nakon testiranja brzine samog algoritma možda uvedemo neke dodatne parametre kako bi ubrzali cijeli proces, ali to zasad nije učinjeno. Jedan od mogućih načina ubrzavanja procesa je dodavanje atributa učestalosti svih kocki (atribut tipa integer koji će se izračunati kao „broj setova u kojima se nalazi određena kockica / ukupan broj setova u bazi“. Nakon toga bi se uspoređivanje provodilo na način da se prvo provjeravaju kockice manje učestalosti, budući da kocke velike učestalosti u manje slučajeva dovode do smanjivanja podskupa rješenja. Samo izračunavanje atributa učestalosti obavlja se samo prilikom ažuriranja baze te nije operacija koja će dodatno opterećivati i usporavati sustav. Ostale funkcije samog sustava svode se na CRUD operacije nad bazom podataka i filtriranja i sortiranja samih podataka prije prikaza korisniku. Kod evidencije korisnika u bazu smo pohranjivali nešifrirane lozinke. Ako bi sustav ikad bio namijenjen za javnost pobrinuli bi se i za sigurnosni aspekt samog sustava.

Baza je implementirana u sustavu SQL Server 2014, međutim napredne funkcionalnosti toga sustava nisu bile potrebne u kreiranju same baze.

Bazu smo kreirali koristeći SQL standard bez proširenja.



Slika . Dijagram relacijske baze podataka

# Use cases

1.   Prijava korisnika na sustav

2. Registracija korisničkog računa

3. Pregled baze svih setova

4. Dodavanje setova u listu želja

5. Dodavanje setova u inventar

6. Dohvat uputa za slaganje setova

7. Pregled sadržaja setova

8. Pregled baze svih kocki

9. Pregled setova s odabranim kockama

10. Pregled vlastite liste želja

11. Uklanjanje setova iz liste želja

12. Pregled vlastitog inventara

13. Uklanjanje setova iz inventara

14. Označavanje setova složenima

15. Dodavanje setova u listu favorita

16. Pregled svih MOC setova

17. Definiranje vlastitog MOC seta

18. Dodavanje popisa kocki u MOC set

19. Pregled vlastite liste favorita

20. Uklanjanje setova iz liste favorita

21. Korištenje pomoćnika za slaganje

22. Pregled korisničkih profila

23. Uređivanje korisničkog profila

## Use case 1: Prijava na sustav

Primary actor: Korisnik

Stakeholders:

Korisnik – Želi se prijaviti na sustav

Preconditions: Korisnik ima pristup sustavu i posjeduje korisnički račun

Postconditions: Korisnik je uspješno prijavljen na sustav

Main success scenario:

1. Korisnik unosi korisničko ime i lozinku

2. Sustav provjerava valjanost unesenih podataka

3. Sustav prihvaća prijavu korisnika

4. Sustav preusmjerava korisnika na početnu stranicu aplikacije

Extensions:

3a. Sustav utvrđuje da korisnički podatci nisu ispravni

        1. Sustav upozorava korisnika na neispravnost podataka

        2. Sustav omogućava korisniku ponovni unos korisničkih podataka

## Use case 2: Registracija korisničkog računa

Primary actor: Korisnik

Stakeholders:

Korisnik – Želi otvoriti korisnički račun

Preconditions: Korisnik ima pristup sustavu

Postconditions: Korisnik je uspješno otvorio korisnički račun

Main success scenario:

1. Korisnik unosi željeno korisničko ime i lozinku

2. Sustav provjerava valjanost unesenih podataka

3. Sustav stvara novi korisnički račun s primljenim podatcima

4. Sustav preusmjerava korisnika na stranicu prijave na sustav

Extensions:

3a. Korisnik s unesenim korisničkim imenom već postoji u sustavu

        1. Sustav upozorava korisnika na zauzetost željenog korisničkog imena

        2. Sustav korisniku omogućuje odabir drugog korisničkog imena

3b. Korisnički podatci su neispravnog formata

        1. Sustav obavještava korisnika o neispravnosti unešenih podataka

        2. Sustav korisniku omogućuje izmjenu unešenih podataka

## Use case 3: Pregled baze svih setova

Primary actor: Korisnik

Stakeholders:

Korisnik – Želi pregledavati bazu setova

Preconditions: Korisnik je uspješno prijavljen na sustav

Postconditions: Korisniku je prikazan popis setova

Main success scenario:

1. Korisnik odabire pregled baze setova

2. Korisnik definira kriterije pretrage

3. Korisnik šalje upit s definiranim kriterijima

4. Sustav prikazuje rezultate pretrage

5. Korisnik sortira dobivene rezultate po proizvoljnom kriteriju

Extensions:

5a. Sustav nije pronašao rezultate za definirane kriterije

1. Korisnik može ponoviti pretragu s novim kriterijima

## Use case 4: Dodavanje setova u listu želja

Primary actor: Korisnik

Stakeholders:

Korisnik – Želi dodati setove u listu želja

Preconditions: Korisnik pregledava popis setova dobiven pretragom baze

Postconditions: Korisnik je uspješno dodao setove u listu želja

Main success scenario:

1. Korisnik odabire jedan ili više setova s popisa

2. Korisnik unosi željenu količinu za svaki odabrani set

3. Korisnik odabire naredbu za dodavanje setova u listu želja

4. Sustav dodaje odabrane setove u listu želja

5. Sustav obavještava korisnika o uspješnom dodavanju setova u listu želja

Extensions:

4a. Sustav utvrđuje da se neki od odabranih setova već nalaze u listi želja

        1. Sustav obavještava korisnika o setovima koji se već nalaze u listi želja

        2. Korisnik potvrđuje koje od tih setova svejedno želi dodati u listu želja

        3. Sustav uvećava količinu odabranih setova u listi želja

## Use case 5: Dodavanje setova u inventar

Primary actor: Korisnik

Stakeholders:

Korisnik – Želi dodati setove u inventar

Preconditions: Korisnik pregledava popis setova dobiven pretragom baze

Postconditions: Korisnik je uspješno dodao setove u inventar

Main success scenario:

1. Korisnik odabire jedan ili više setova s popisa

2. Korisnik unosi željenu količinu za svaki od odabranih setova

3. Korisnik odabire naredbu za dodavanje setova u inventar

4. Sustav odabrane setove uklanja s liste želja

5. Sustav obavještava korisnika o uspješnom dodavanju setova u inventar

Extensions:

4a. Sustav utvrđuje da se neki od odabranih setova ne nalaze u listi želja

        1. Sustav iz liste želja uklanja samo one setove koji se u njoj nalaze

4b. Sustav utvrđuje da je količina nekih od odabranih setova u listi želja veća od količine tih setova koja se dodaje u inventar

1. Sustav umanjuje količinu tih setova u listi želja za količinu koja se dodaje u inventar

## Use case 6: Dohvat uputa za slaganje setova

Primary actor: Korisnik

Stakeholders:

Korisnik – Želi dohvatit upute za slaganje setova

Preconditions: Korisnik pregledava popis setova dobiven pretragom baze, sustav ima pristup internetu

Postconditions: Upute za slaganje su pohranjene na uređaju

Main success scenario:

1. Korisnik odabire jedan ili više setova s popisa

2. Korisnik odabire naredbu za dohvat uputa za slaganje

3. Sustav dohvaća odabrane upute za slaganje

4. Sustav korisniku omogućava odabir lokacije za pohranu uputa

5. Korisnik odabire lokaciju na uređaju gdje želi pohraniti upute

6. Sustav pohranjuje upute na zadanu lokaciju

Extensions:

3a. Sustav ne može dohvatiti željene upute

        1. Sustav obavještava korisnika da ne može dohvatiti upute

6a. Sustav ne može pohraniti upute na zadanu lokaciju

        1. Sustav obavještava korisnika da ne može pohraniti upute

1. Sustav korisniku omogućuje odabir druge lokacije za pohranu

## Use case 7: Pregled sadržaja setova

Primary actor: Korisnik

Stakeholders:

Korisnik – Želi pregledati sadržaj setova

Preconditions: Korisnik pregledava popis setova dobiven pretragom baze

Postconditions: Korisniku se prikazuje popis kocki sadržanih u setovima

Main success scenario:

1. Korisnik odabire jedan ili više setova s popisa

2. Korisnik odabire naredbu za prikaz sadržaja setova

3. Sustav prikazuje popis svih kocki sadržanih u odabranim setovima

4. Korisnik sortira popis kocki po proizvoljnom kriteriju

## Use case 9: Pregled setova s odabranim kockama

Primary actor: Korisnik

Stakeholders:

Korisnik – Želi vidjeti koji setovi sadrže sve odabrane kocke

Preconditions: Korisnik pregledava popis kocki dobiven pretragom baze

Postconditions: Korisniku je prikazan popis setova

Main success scenario:

1. Korisnik odabire jednu ili više kocki

2. Korisnik unosi željenu količinu za svaku od odabranih kocki

3. Korisnik odabire naredbu za pregled setova s odabranim kockama

4. Sustav prikazuje popis svih setova koji sadrže odabrane kocke

5. Korisnik sortira dobivene rezultate po proizvoljnom kriteriju

Extensions:

5a. Sustav nije pronašao rezultate za definirane kriterije

1. Korisnik može ponoviti pretragu s novim kriterijima

## Use case 12: Pregled vlastitog inventara setova

Primary actor: Korisnik

Stakeholders:

Korisnik – Želi pregledavati svoj inventar

Preconditions: Korisnik je uspješno prijavljen na sustav

Postconditions: Korisniku se prikazuje popis setova u njegovom inventaru

Main success scenario:

1. Korisnik odabire pregled vlastitog inventara

2. Korisnik definira kriterije pretrage

3. Korisnik šalje upit s definiranim kriterijima

4. Sustav prikazuje rezultate pretrage

5. Korisnik sortira dobivene rezultate po proizvoljnom kriteriju

Extensions:

5a. Sustav nije pronašao rezultate za definirane kriterije

1. Korisnik može ponoviti pretragu s novim kriterijima

## Use case 13: Uklanjanje setova iz inventara

Primary actor: Korisnik

Stakeholders:

Korisnik – Želi ukloniti setove iz inventara

Preconditions: Korisnik pregledava popis setova u svom inventaru

Postconditions: Korisniku je uspješno uklonio set iz inventara

Main success scenario:

1. Korisnik odabire jedan ili više setova s popisa

2. Korisnik unosi željenu količinu za svaki od odabranih setova

3. Korisnik odabire naredbu za uklanjanje setova iz inventara

4. Sustav uklanja odabrane setove iz inventara

5. Sustav obavještava korisnika o uspješnom uklanjanju setova iz inventara

Extensions:

4a. Sustav utvrđuje da se u inventaru nalazi manja količina setova nego što želi ukloniti

1. Sustav obavještava korisnika da nije moguće ukloniti veću količinu setova nego što se nalazi u inventaru

## Use case 14: Označavanje setova složenima

Primary actor: Korisnik

Stakeholders:

Korisnik – Želi označiti setove složenima

Preconditions: Korisnik pregledava popis setova u svom inventaru

Postconditions: Korisniku je uspješno označio setove

Main success scenario:

1. Korisnik odabire jedan ili više setova s popisa

2. Korisnik unosi željenu količinu za svaki od odabranih setova

3. Korisnik odabire naredbu za označavanje setova složenima

4. Sustav označava odabrane setove

Extensions:

4a. Sustav utvrđuje da se u inventaru nalazi manja količina setova nego što se želi označiti

1. Sustav obavještava korisnika da nije moguće označiti veću količinu setova nego što se nalazi u inventaru

## Use case 17: Definiranje vlastitog MOC seta

Primary actor: Korisnik

Stakeholders:

Korisnik – Želi definirati vlastiti set

Preconditions: Korisnik je uspješno prijavljen na sustav

Postconditions: Korisnik je uspješno definirao vlastiti set

Main success scenario:

1. Korisnik odabire definiranje vlastitog seta

2. Korisnik unosi željeni naziv seta

3. Korisnik unosi opis seta u tekstualnom obliku

4. Korisnik dodaje popis kocki potrebnih za slaganje seta

5. Korisnik dodaje upute za slaganje seta

6. Korisnik dodaje fotografije složenog seta

7. Korisnik odabire naredbu za dodavanje seta u sustav

8. Sustav popunjava ostale podatke o setu

9. Sustav dodaje set u bazu MOC setova

10. Sustav obavještava korisnika o uspješnom dodavanju seta

Extensions:

4a. Korisnik ne dodaje popis kocki potrebnih za slaganje seta

        1. Korisnik nastavlja sa sljedećim korakom

5a. Korisnik ne dodaje upute za slaganje seta

        1. Korisnik nastavlja sa sljedećim korakom

6a. Korisnik ne dodaje fotografije složenog seta

1. Korisnik nastavlja sa sljedećim korakom

## Use case 18: Dodavanje popisa kocki u MOC set

Primary actor: Korisnik

Stakeholders:

Korisnik – Želi dodati popis kocki potrebnih za slaganje u set

Preconditions: Korisnik definira vlastiti MOC set

Postconditions: Korisnik je uspješno dodao popis kocki

Main success scenario:

1. Korisnik odabire naredbu za dodavanje popisa kocki

2. Sustav omogućava korisniku dodavanje popisa kocki

3. Korisnik pregledava bazu svih kocki

4. Korisnik odabire jednu ili više potrebnih kocki

5. Korisnik unosi potrebnu količinu za svaku od odabranih kocki

6. Korisnik odabire naredbu za dodavnje odabranih kocki u popis potrebnih kocki

7. Sustav dodaje odabrane kocke u popis kocki

8. Korisnik ponavlja korake 2. – 4. dok ne doda sve potrebne kocke

9. Korisnik odabire naredbu za dodavanje popisa kocki u set

10. Sustav dodaje popis kocki u set

11. Sustav obavještava korisnika o uspješnom dodavanju kocki u set

12. Sustav omogućava korisniku nastavak definiranja seta

Extensions:

7a. Sustav utvrđuje da neke od odabranih kocki već postoje u popisu potrebnih kocki

        1. Sustav obavještava korisnika o kockama koje već postoje u popisu

        2. Korisnik potrvđuje koje od tih kocki svejedno želi dodati u popis kocki

        3. Sustav uvećava količinu odabranih kocki u popisu kocki

8a. Korisnik želi ukloniti jednu ili više kocki s popisa

        1. Korisnik odabire sve kocke koje želi ukloniti

        2. Korisnik odabire naredbu za uklanjanje odabranih kocki

        3. Sustav uklanja odabrane kocke s popisa

## Use case 21: Korištenje pomoćnika za slaganje

Primary actor: Korisnik

Stakeholders:

Korisnik – Želi saznati koje setove može složiti s kockama koje ima na raspolaganju

Preconditions: Korisnik je uspješno prijavljen na sustav

Postconditions: Korisnik je dobio prijedloge setova koje može složiti

Main success scenario:

1. Korisnik odabire korištenje pomoćnika za slaganje

2. Korisnik odabire korištenje svih kocki koje posjeduje

3. Korisnik definira kriterije pretrage

4. Korisnik odabire naredbu za slanje upita

5. Sustav prikazuje popis setova koje korisnik može složiti

6. Korisnik sortira rezultate po proizvoljnom kriteriju

Extensions:

2a. Korisnik odabire korištenje samo onih kocki koje trenutno nisu iskorištene za slaganje nekog drugog seta

6a. Sustav nije pronašao setove koji zadovoljavaju korisnikove kriterije

1. Korisnik može ponoviti pretragu s drugim kriterijima

## Use case 22: Pregled korisničkih profila

Primary actor: Korisnik

Stakeholders:

Korisnik – Želi pregledavati profile drugih korisnika sustava

Preconditions: Korisnik je uspješno prijavljen na sustav

Postconditions: Korisnik je dobio prikaz željenih profila

Main success scenario:

1. Korisnik odabire pregled korisničkih profila

2. Sustav prikazuje popis korisničkih profila

3. Korisnik odabire profil koji želi pregledati

4. Sustav korisniku prikazuje odabrani profil

5. Korisnik pregledava prikazani profil

6. Korisnik završava s pregledavanjem profila

7. Sustav uklanja prikaz profila

## Use case 23: Uređivanje korisničkog profila

Primary actor: Korisnik

Stakeholders:

Korisnik – Želi promjeniti osobne podatke u sustavu

Preconditions: Korisnik je uspješno prijavljen na sustav

Postconditions: Korisnik je uspješno promjenio svoje podatke

Main success scenario:

1. Korisnik odabire uređivanje korisničkog profila

2. Sustav omogućuje korisniku izmjenu korisničkih podataka

3. Korisnik upisuje nove korisničke podatke

4. Sustav pohranjuje promjene

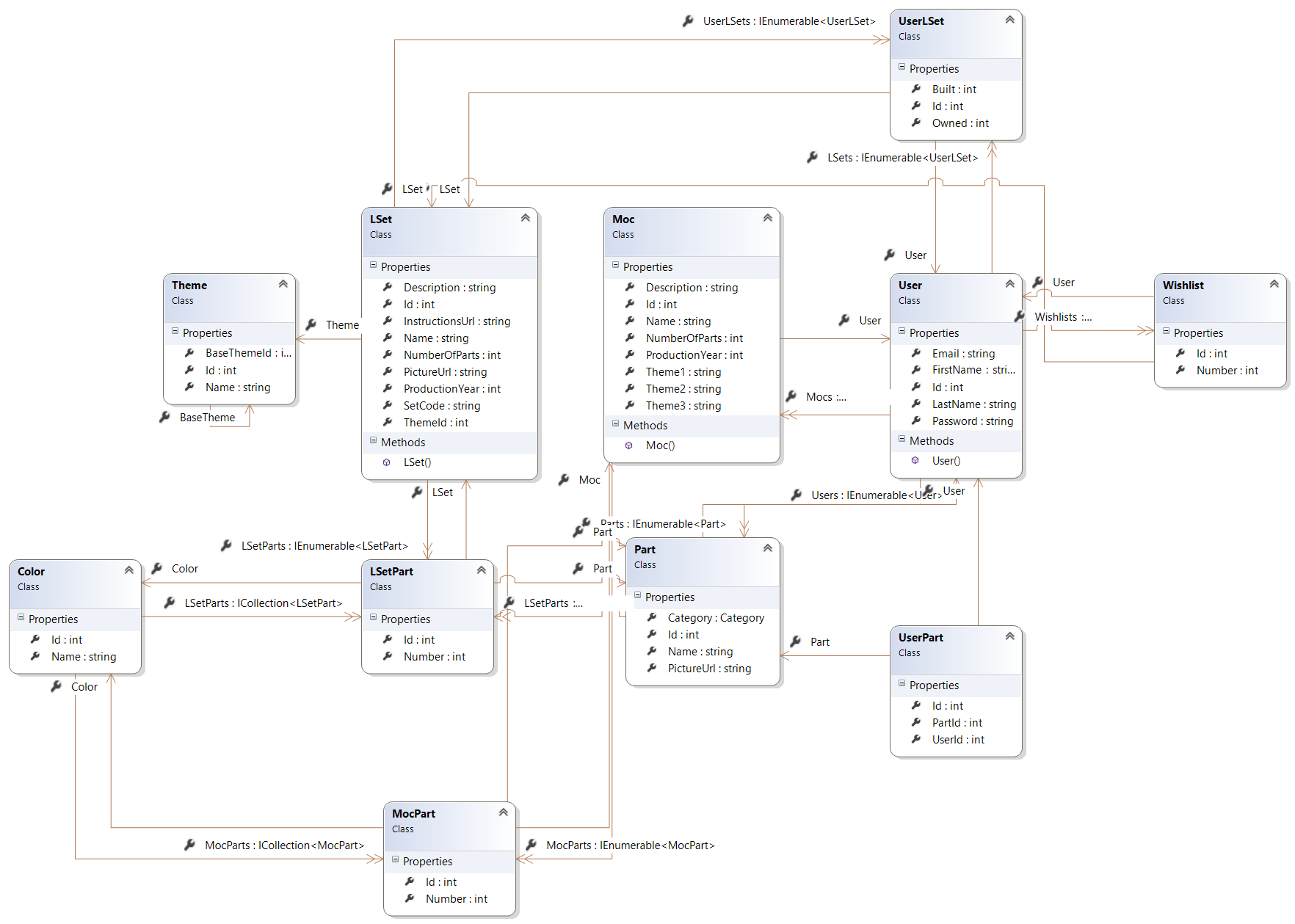
Extensions:

4a. Sustav utvrđuje da unešeni podaci nisu odgovarajućeg formata

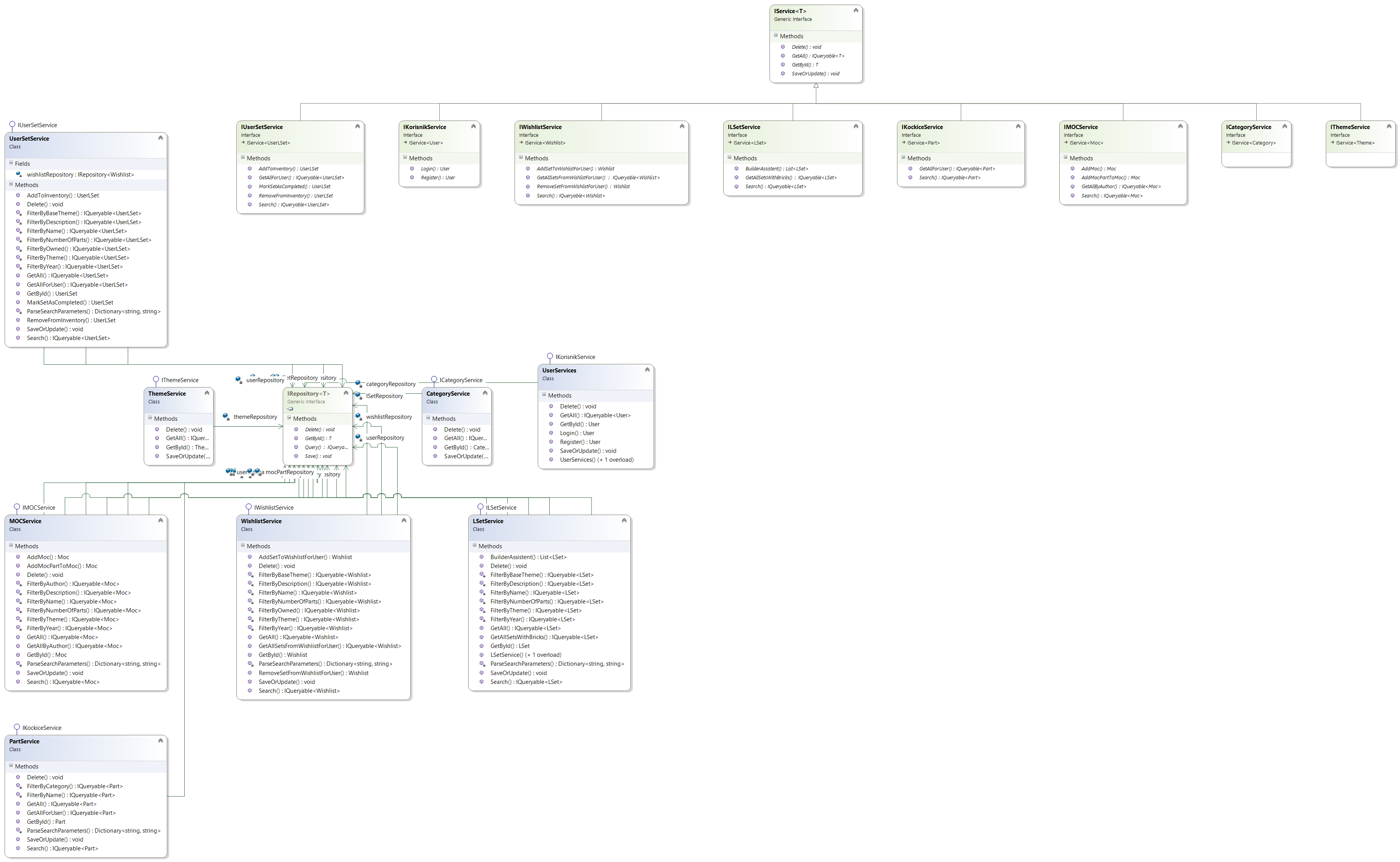
        1. Sustav obavještava korisnika o neispravnosti podataka

        2. Sustav omogućuje korisniku ponovni unos podataka

# Opis objektnog modela



Slika . Dijagram klasa domenskog modela



Slika . Dijagram repozitorija i servisa

Entiteti - User, LSet, Moc, Part, Color, Category, Theme

Repozitoriji - repozitoriji su izvedeni generički tako da nam svaki entitet može imati svoj repozitorij

Servisi - Svi servisi nasljeđuju bazno sučelje servisa, te implementiraju dodatne metode specifične za pojedini servis.

# Opis implementacije perzistencije

Repoziroriji su izvedeni generički preko sučelja IRepository<T> koje onda nasljeđuje klasa Repository<T>. Nalaze se u Data sloju te im je pri inicijalizaciju moramo predati enitiet na koji se taj repozitorij korisit. Transkacije su implementirane tako da spremamo transakciju u privatnu varijablu repozitorija nad kojim se vrši transakcija te možemo izvršiti standardne naredbe BeginTransaction(), CommitTransaction(), RollbackTransaction(), CloseTransaction().

Mapiranje je izvedeno pomoću Fluent NHibernate knjižnice.

Konfiguracija mapiranja se nalazi u klasi Database.cs, te u metodi SessionFactory postavljamo parametre spajanja na bazu podataka te assembly za mapiranje.

private static ISessionFactory SessionFactory

{

get

{

return Fluently.Configure()

.Database(

MsSqlConfiguration.MsSql2012.ConnectionString(c => c.FromConnectionStringWithKey("ConnectionString"))

)

.Mappings(m =>

m.FluentMappings.AddFromAssemblyOf<User>())

.BuildSessionFactory();

}

}

Postavljeno je tako da se koristi SQL Server, te preko Assemlya tipa User možemo dohvatiti sve ostale mape za koje je potrebno izvesti mapiranje.

Primjer mapiranje najvažnijh klasa:

Klasa User:

public class UserMap : ClassMap<User>

{

public UserMap()

{

Table("users");

Id(x => x.Id).Column("id").GeneratedBy.Native();

Map(x => x.Email).Column("email");

Map(x => x.Password).Column("password");

Map(x => x.FirstName).Column("first\_name");

Map(x => x.LastName).Column("last\_name");

HasManyToMany(x => x.Parts).Cascade.All().Table("user\_part");

HasMany(x => x.LSets).Cascade.All().Table("user\_Lset");

HasMany(x => x.Mocs).Cascade.All().Table("user\_MOC");

HasMany(x => x.Wishlists).Cascade.All().Table("wishlist");

}

}

Klasa LSet:

public class LSetMap : ClassMap<LSet>

{

public LSetMap()

{

Table("Lset");

Id(x => x.Id).Column("id").GeneratedBy.Native();

Map(x => x.Name).Column("name");

Map(x => x.ProductionYear).Column("production\_year");

Map(x => x.NumberOfParts).Column("num\_parts");

Map(x => x.Description).Column("description");

Map(x => x.ThemeId).Column("id\_theme");

Map(x => x.InstructionsUrl).Column("instructions\_url");

Map(x => x.PictureUrl).Column("picture\_url");

References(x => x.Theme).Column("id\_theme");

HasMany(x => x.UserLSets).Table("user\_Lset");

HasMany(x => x.LSetParts).Cascade.All().Table("Lset\_parts");

}

}

Klasa UserLSet

public class UserLSetMap : ClassMap<UserLSet>

{

public UserLSetMap()

{

Table("user\_Lset");

Id(x => x.Id).Column("id").GeneratedBy.Native();

Map(x => x.Built).Column("built");

Map(x => x.Owned).Column("owned");

References(x => x.User).Column("id\_user");

References(x => x.LSet).Column("id\_Lset");

}

}

Klasa Part

public class PartMap : ClassMap<Part>

{

public PartMap()

{

Table("part");

Id(x => x.Id).Column("id").GeneratedBy.Native();

Map(x => x.Name).Column("name");

Map(x => x.PictureUrl).Column("picture\_url");

References(x => x.Category).Column("category");

HasManyToMany(x => x.Users)

.Cascade.All()

.Inverse()

.Table("user\_part");

HasMany(x => x.LSetParts)

.Cascade.All()

.Inverse()

.Table("Lsets\_parts");

}

}

Klasa Moc:

public class MocMap : ClassMap<Moc>

{

public MocMap()

{

Table("MOC");

Id(x => x.Id).Column("id").GeneratedBy.Native();

Map(x => x.Name).Column("name");

Map(x => x.NumberOfParts).Column("num\_parts");

Map(x => x.Theme1).Column("theme1");

Map(x => x.Theme2).Column("theme2");

Map(x => x.Theme3).Column("theme3");

Map(x => x.Description).Column("description");

Map(x => x.AuthorId).Column("id\_author");

Map(x => x.ProductionYear).Column("production\_year");

References(x => x.UserMoc).Column("id\_author").ReadOnly();

HasMany(x => x.MocParts).Cascade.All().Table("MOC\_parts");

}

}

# Opis izgrađene desktop aplikacije

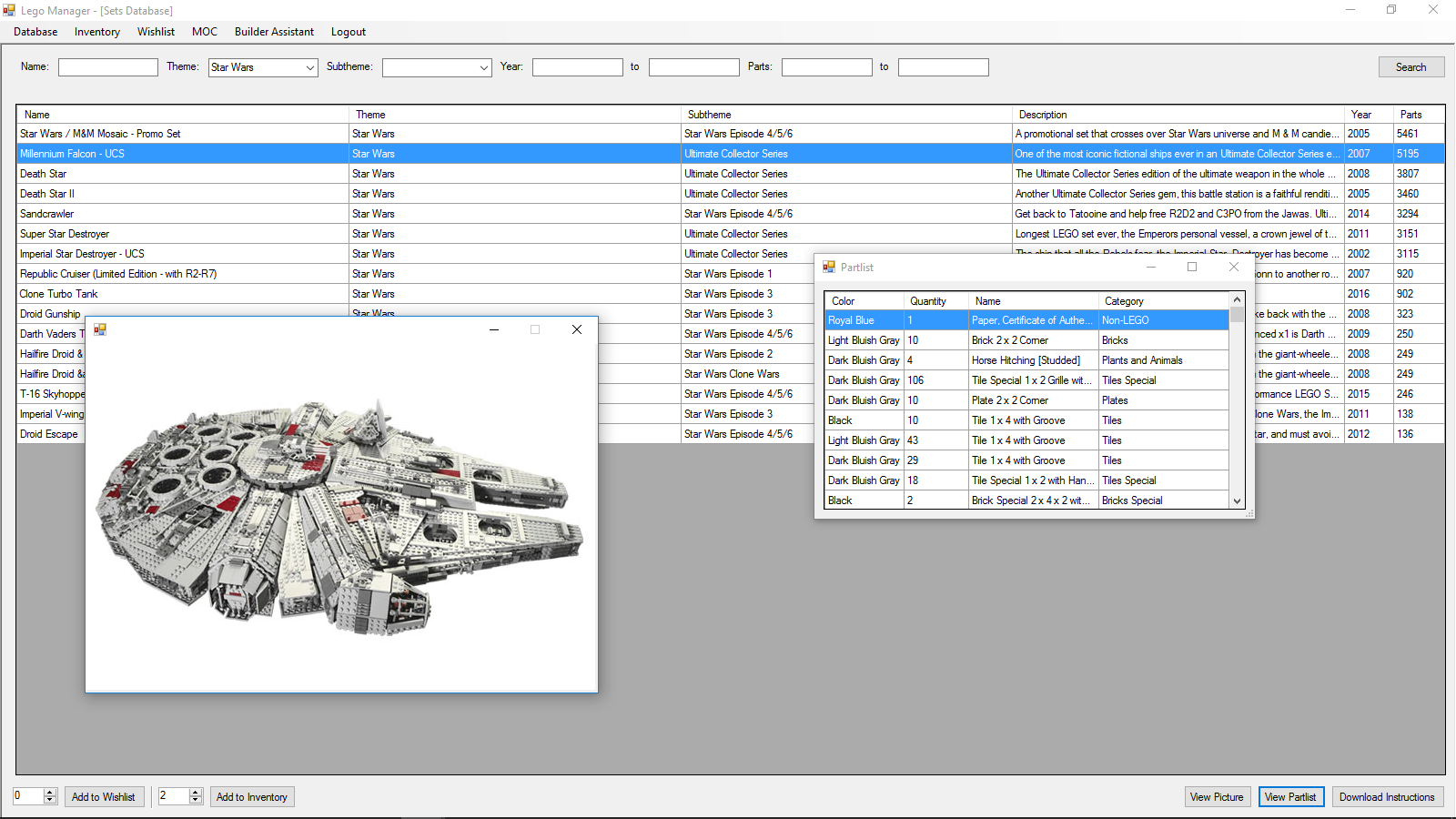
Za izradu desktop aplikacije korišten je programski jezik C#, te razvojno okruženje Microsoft Visual Studio. Grafičko korisničko sučelje ostvareno je pomoću .NET Frameworka, kokretno Windows Forms-a.

Izvorni kod organiziran je u nekoliko direktorija. U direktoriju BaseLib nalaze se sučelja svih Viewova, kao i sučelje tvornice za njihovo stvaranje. Implementacije svih navedenih sučelja se nalaze u direktoriju Views. Naposlijetku, u direktoriju Controllers se nalaze Controlleri(Presenteri) za sve Viewove.

Prilikom izrade aplikacije korišten je MVP obrazac, i to u varijanti pasivnog Viewa. Jedina zadaća pasivnih Viewova je registriranje događaja te prosljeđivanje poziva Controlleru. Controller pristupa kontrolama Viewa putem njegovog sučelja, te mu nije poznata konkretna implementacija. Putem navedenog sučelja Controller dohvaća podatke unesene od strane korisnika, odnosno osvježava prikaz podataka i kontrola nakon promjena u modelu. Kontroler također obavlja osnovnu obradu podataka, poput verifikacije korisničkog unosa ili pripreme podataka za prikaz, dok se za sve složenije funkcionalnosti oslanja na servise.

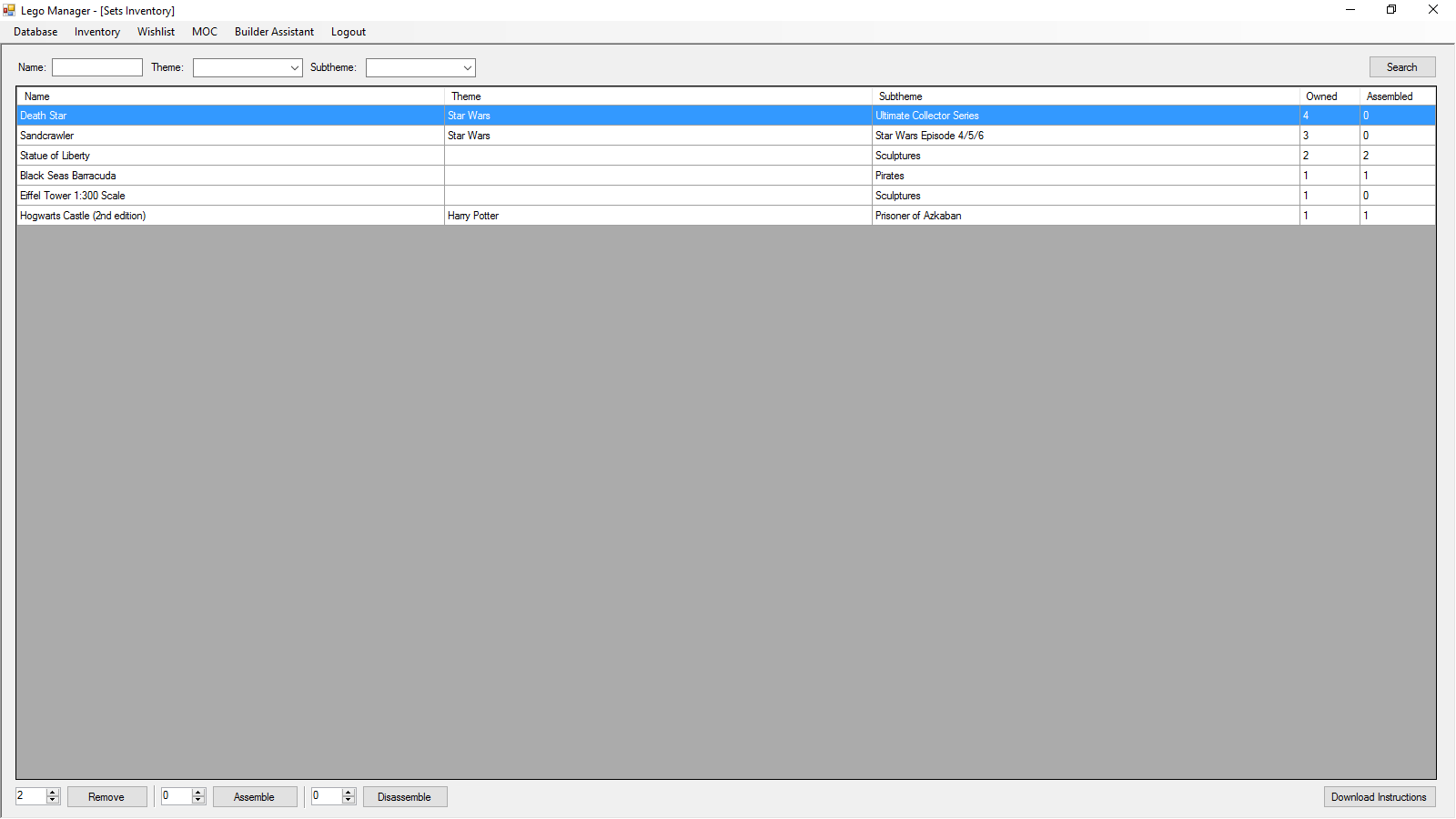
Ulazna točka aplikacije je forma za prijavu korisnika, u kojoj se od korisnika traži njegova E-mail adresa i lozinka. Ukoliko korisnik nema račun, može odabrati gumb za registraciju koji ga vodi na novu formu u kojoj upisuje svoje podatke i stvara korisnički račun.

Nakon uspješne prijave, korisnik na raspolaganju ima više opcija u sklopu glavnog izbornika. Može pregledavati bazu setova ili kocki, kao i vlastiti inventar te listu želja. Ponuđene su mu i aktivnosti vezane uz setove definirane od strane korisnika (MOC - My Own Creation), te pomoćnik za slaganje (Builder Assistant).



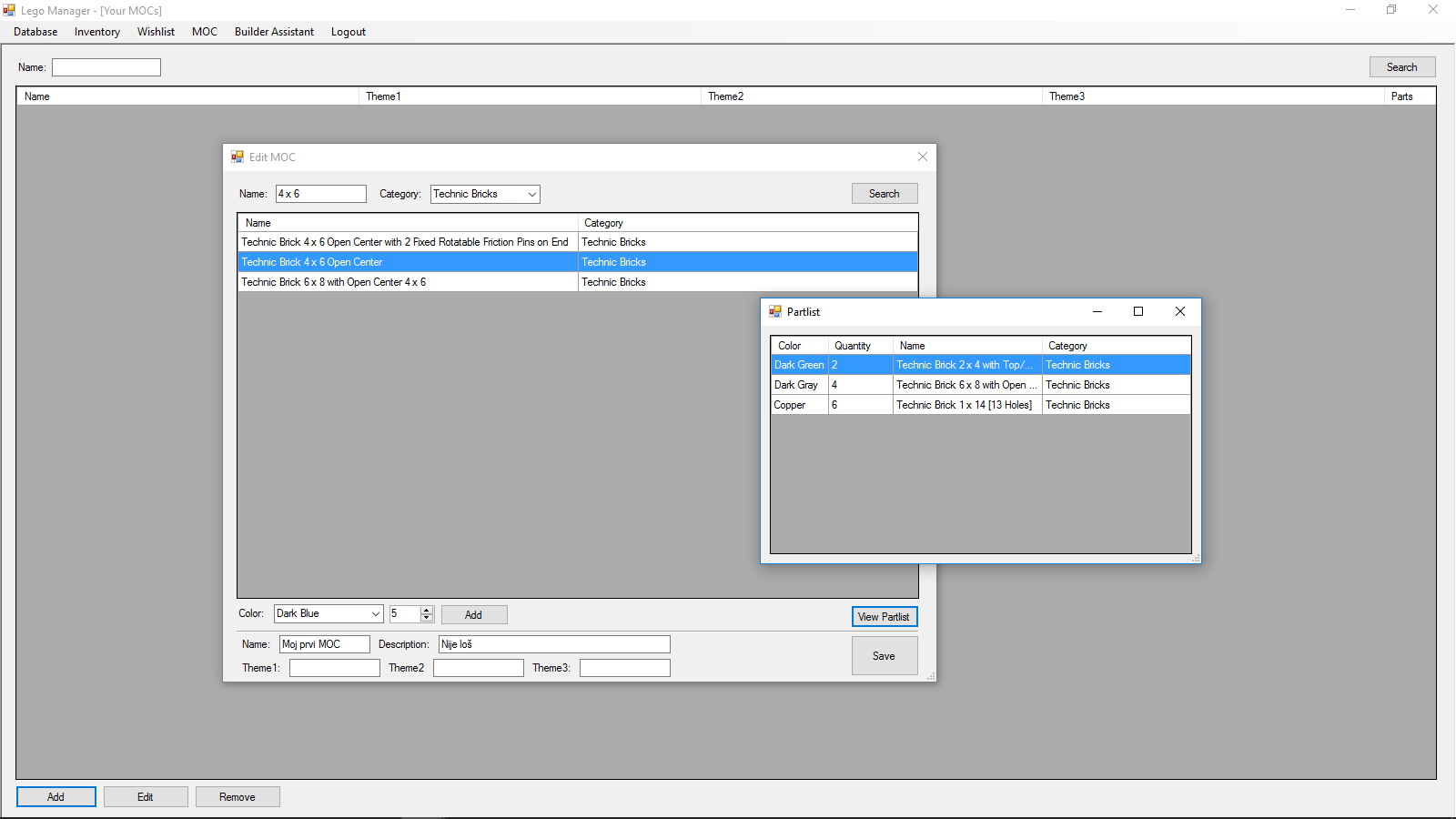
Slika .Pregledavanje baze setova

Prilikom pregledavanja baze setova korisnik na raspolaganju ima pretragu po raznim kriterijima (naziv, tema, podtema, godina izdavanja, broj dijelova). Odabrane setove može dodavati u svoj inventar ili listu želja, a nude mu se i opcije za prikaz slike, popisa dijelova sadržanih u setu i uputa za slaganje. Za prikaz slike i uputa potreban je pristup internetu. Kod pregledavanja baze kocki, također je dostupna pretraga (naziv, kategorija), a moguće je i dobiti popis setova u kojima su sadržane kocke odabrane s popisa.



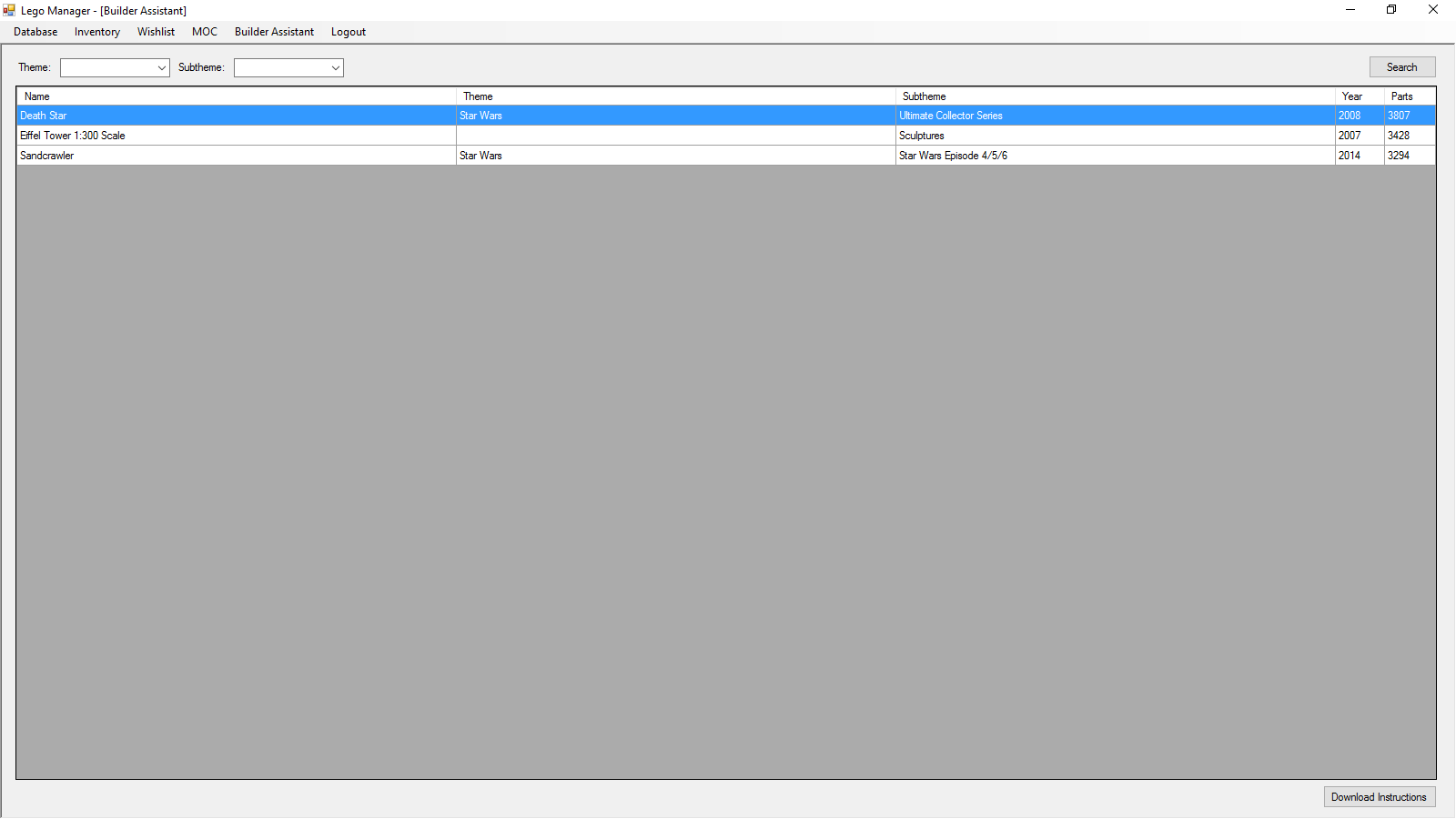
Slika . Pregledavanje inventara

U svom inventaru korisniku se uz setove prikazuju podaci o broju primjeraka u njegovom vlasništvu, kao i o broju trenutno složenih primjeraka. Korisnik odabrani set može ukloniti iz inventara, označiti ga složenim, ili ga označiti rastavljenim. Korisnikova lista želja je slična inventaru, s razlikom da je kod nje na raspolaganju samo uklanjanje setova. U oba slučaja korisnik na raspolaganju ima pretragu (naziv, tema, podtema).



Slika . Definiranje vlastitog seta

Osim što može pregledavati setove definirane od strane drugih korisnika, korisnik može i dodavati, uređivati i brisati vlastite setove. Prilikom definiranja seta korisnik je obavezan setu dati naziv i dodati popis kocaka potrebnih za njegovo slaganje. Prilikom dodavanja kocki, osim količine potrebno je odabrati i njihovu boju, a u svakom trenutku moguće je pogledati popis već dodadnih kocaka. Opcionalno korisnik može unjeti kratki opis seta, te do tri teme koje ga opisuju.



Slika . Pomoć pri pronalasku seta

Naposlijetku, korisniku se nudi pomoć pri pronalasku setova koje može složiti koristeći kocke koje već ima u svojem vlasništvu. U obzir se uzimaju samo one kocke koje trenutno nisu iskorištene za slaganje nekog drugog seta, a dobivene rezultate moguće je dodatno filtrirati (tema, podtema).

# Opis izgrađene web aplikacije

Web aplikacija izgrađena je korištenjem C# jezika i ASP.NET frameworka. Struktura aplikacije odgovara MVC patternu. Većina programskog koda smještena je u datotekama Controllers, Models i Views. Dio datotečne strukture prikazan je na Slici 1.

Kontroleri:

AccountController klasa služi za upravljanje prijavom i registracijom, pri registraciji se pozivanjem User servisa stvara novi korisnik, pri prijavi u aplikaciju se preko istog servisa dolazi do User data modela koji odgovara prijavljenom korisniku i sprema ga se u HttpCookie, a pri odjavi se taj cookie briše.

FormsAuthenticationTicket authTicket = new FormsAuthenticationTicket(

1,

model.Email,

DateTime.Now,

DateTime.Now.AddMinutes(15),

false,

userData);

string encTicket = FormsAuthentication.Encrypt(authTicket);

HttpCookie faCookie = new HttpCookie(FormsAuthentication.FormsCookieName, encTicket);

Response.Cookies.Add(faCookie);

Korisniku se nakon uspješne prijave ili odjave prikazuje početni ekran, a nakon registracije preusmjerava ga se na login.

BlockController klasa upravlja kockicama. Korištenjem LSet servisa dolazi se do dijelova setova i detalja o njima koji se prosljeđuju odgovarajućim View-ovima.

CommunityController koristeći Users servis prikazuje View s popisom svih korisnika sustava ili detaljan profil pojedinog korisnika.

HomeController vraća početnu stranicu bez ikakvih podataka.

InventoryController upravlja korisnikovim inventarom, prikazuje sve setove iz inventara i procesira pretraživanje i prikaz rezultata inventara setova. U njemu se također nalaze AJAX metode za dodavanje/micanje setova u inventar i dodavanje/micanje složenih setova.

SearchController upravlja pretraživanjem baze svih setova i kockica.

SetController služi za dohvaćanje detalja o pojedinom setu.

WishlistController upravlja korisnikovom listom želja, dohvaća podatke o listi, procesira pretraživanje i AJAX zahtjeve za dodavanjem/uklanjanjem setova iz liste želja.

U datoteci Helpers nalazi se statična klasa SearchHelper koja ima metodu ConstructSearchParameters koja iz search modela generira string parametara koji search kontroler prosljeđuje odgovarajućem servisu za pretraživanje.

Modeli:

AccountViewModels, CustomPrincipal i CustomPrincipalSerializeModel služe za upravljanje korisnikom pri prijavi i registraciji,

U mapi Interfaces nalazi se ICustomPrincipal interface koji nasljeđuje IPrincipal i njegova implementacija se koristi za upravljanje ulogiranim korisnikom.

SearchBlockModel sadrži atribute za pretraživanje liste kockica.

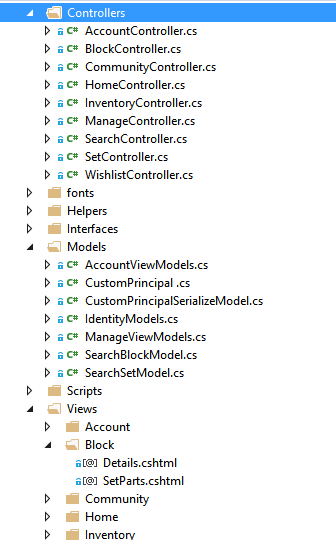
SearchSetModel sadrži atribute za pretraživanje liste setova.

View-ovi:

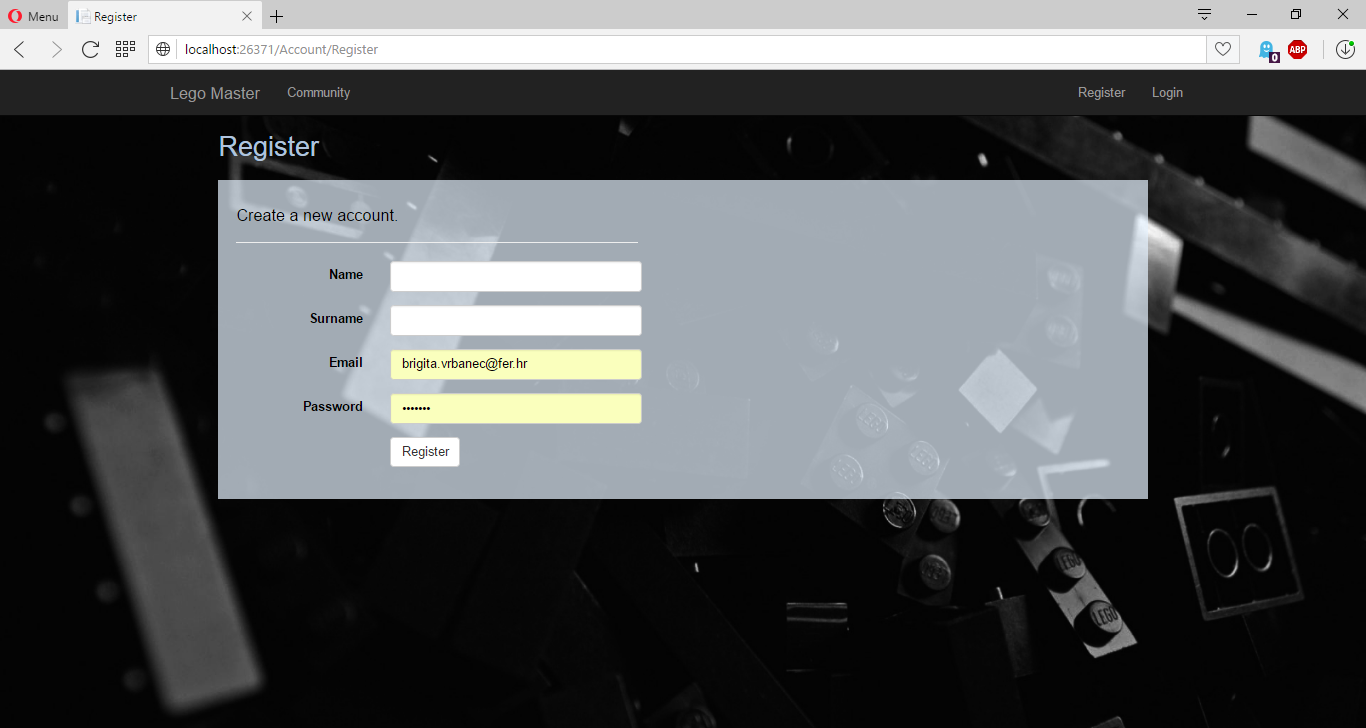
Mapa Views podijeljena je u poddirektorije nazvanje po kontrolerima. Većina jednostavnih view-ova koristi modele iz Data domene za prikaz podataka, a za spremanje parametara pretrage view-ovi koriste odgovarajuće Search modele koje prosljeđuju parcijalnim view-ovima \_blocksSearchPartial i \_setsSearchPartial u Shared mapi. U View-u \_Layout definiran je izgled navigacijske trake.

U mapi Content nalaze se css datoteke za stiliziranje aplikacije, a u mapi Scripts JavaScript skripte.

Datoteka Global.asax sadrži override metode Application\_PostAuthenticateRequest kojom se podaci o trenutno prijavljenom korisniku spremaju u HttpContext.Current.User kako bi im se moglo jednostavno pristupiti iz kontrolera.

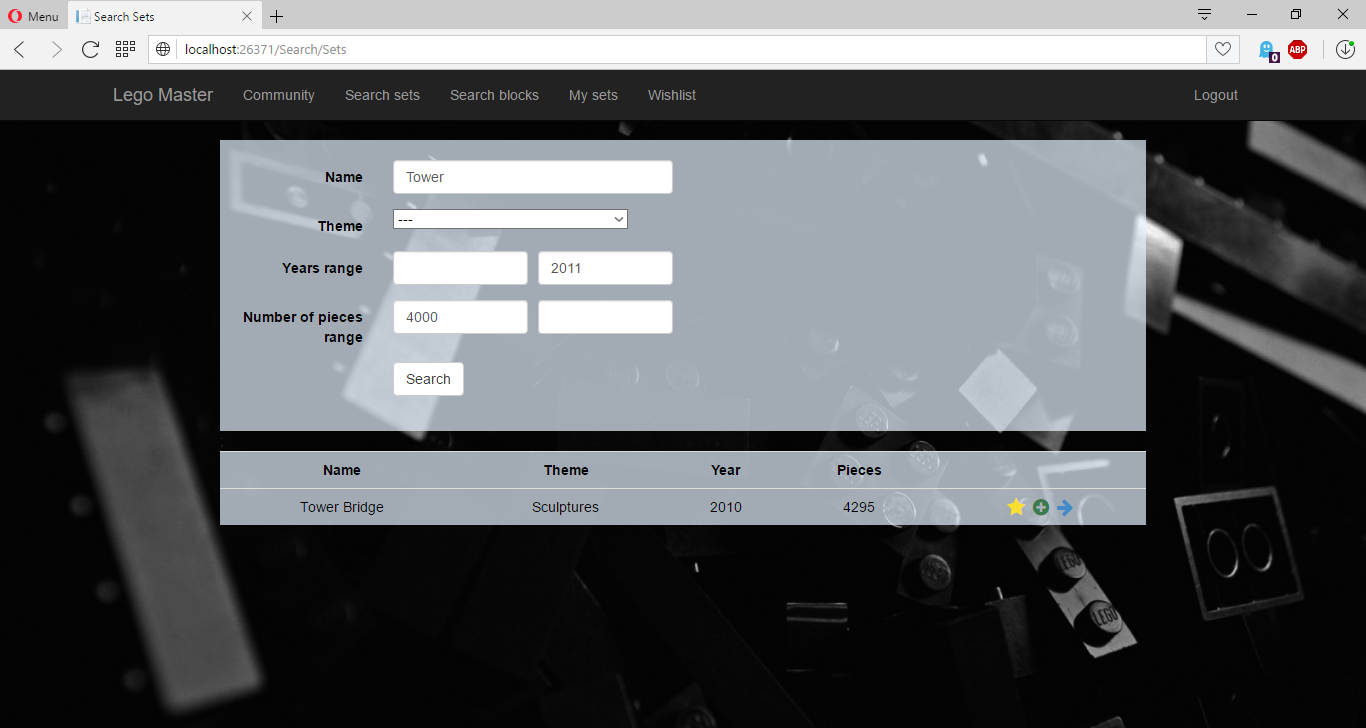


Slika . Struktura datoteka



Slika . Registracija

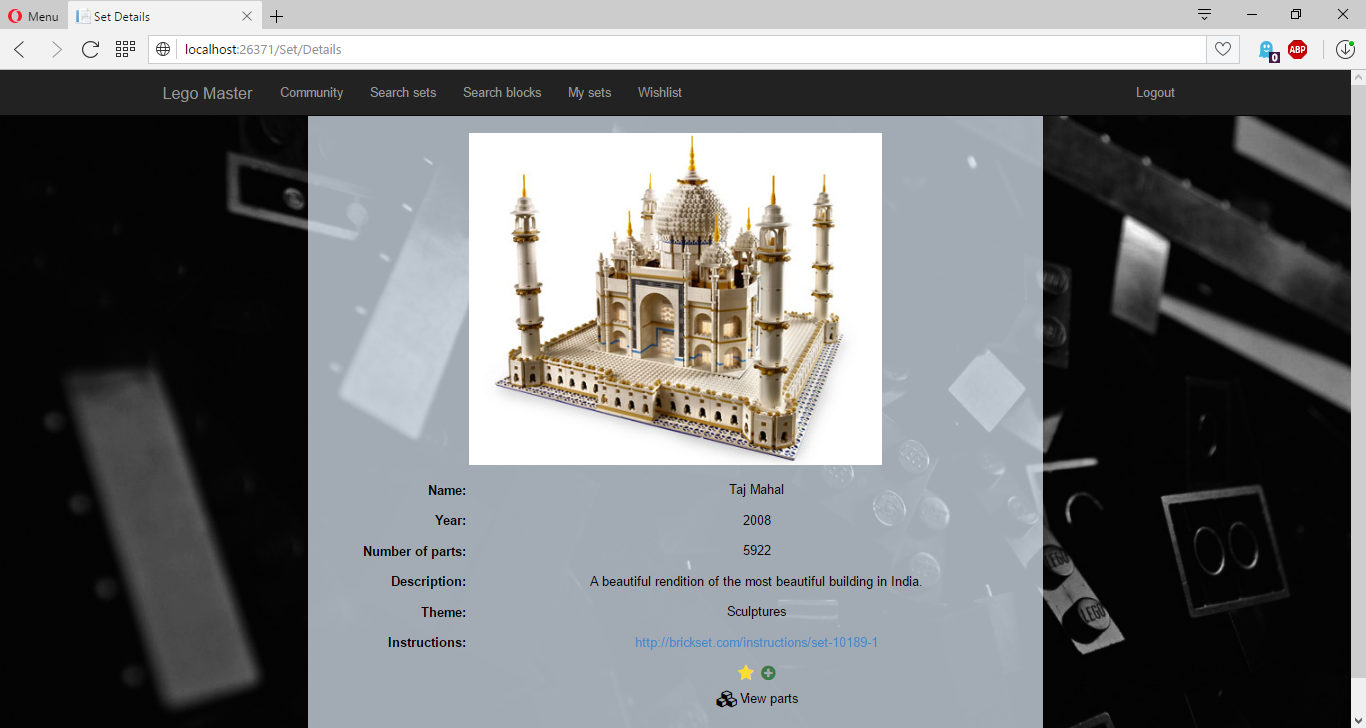
Na slici 9. prikazan je prozor koji se korisniku otvori klikom na Register u navigacijskoj traci. Korisnik mora popuniti sva zadana polja i email adresa mora biti u valjanom formatu.



Slika . Pretraživanje baze setova

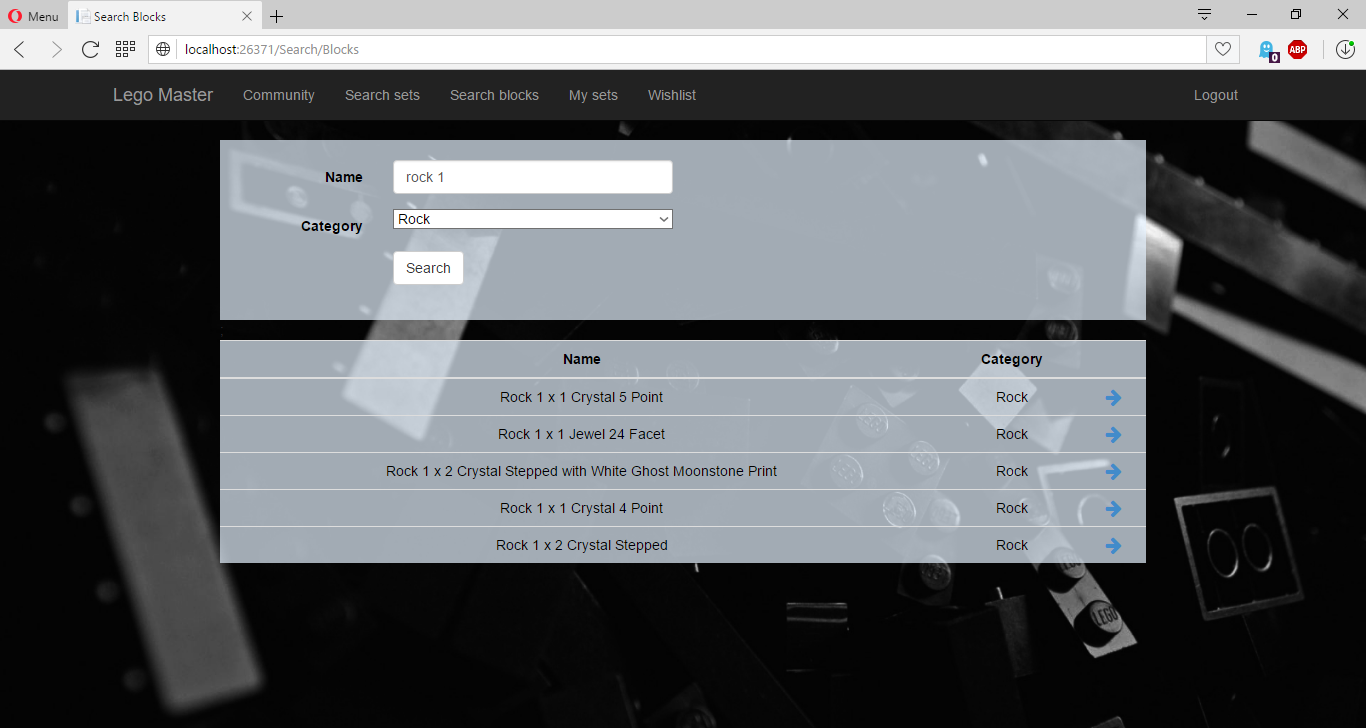
Na slici 10. prikazan je ekran nakon obavljenog pretraživanja baze setova koje je dalo jedan rezultat. Korisnik ne mora unijeti ni jedan parametar, a parametri koje unese se pri pretraživanju spajaju veznikom i (and).

Klikom na žutu zvjezdicu, obavlja se AJAX poziv prema kontroleru koji jedan primjerak seta dodaje u korisnikovu listu želja, a klik na zeleni plus znak dodaje jedan primjerak u korisnikov inventar. Plava strelica vodi na detaljni prikaz seta.



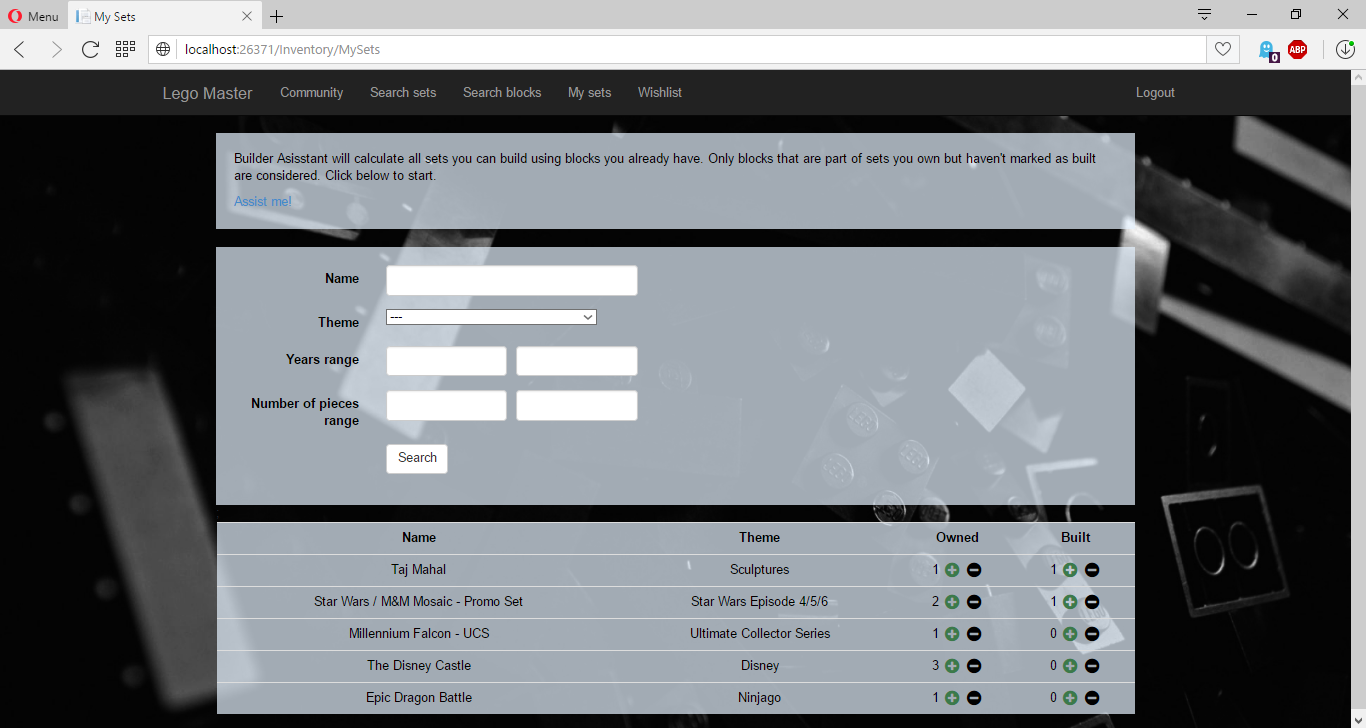
Slika . Prikaz seta

Slika 11. prikazuje jedan set. Ukoliko za set postoji slika, ona se prikaže. Instructions link vodi na vanjsku stranicu na kojoj se nalaze upute za slaganje tog seta. Klikom na View parts otvara se stranica s izlistanim svim dijelovima tog seta.



Slika . Pretraživanje kockica

Na slici 12. prikazan je ekran nakon obavljenog pretraživanja baze kockica koje je dalo više rezultata. Mehanizam pretraživanja je isti kao kod setova, a klik na strelicu vodi na detalje o toj kockici.



Slika . Inventar setova

Na slici 13. Vidi se prikaz korisnikovog inventara setova. Klikom na Assist me! otvara se nova stranica na kojoj su izlistani svi setovi koje korisnik može složiti od dostupnih kockica.

Korisnik može pretraživati svoju listu setova, a klik na + ili - znak povećava ili smanjuje broj setova koje posjeduje ili je izgradio.

# Opis izgrađene mobilne aplikacije

Web Api je realiziran korištenjem ASP.Net Web APi 2 frameworka. Izlazna i ulazna komunikacija na APi se vrši HTTP pozivima na rute za pojedine controllere. Ruta se generira prema pravilima iz klase RouteConfig.cs te je uvijek u obliku: {controller}/{action}/{id}. Gdje nam je controller prefiks riječi Controller u nazivu controllera, action naziv metode koju se poziva, te Id opcionalan. Controlleri se nalaze u mapi Controllers -> Api te uz sufiks Controller u nazivu smo uveli konvenciju da sadrži riječ Api koja je prefiks riječi Controller te sufiks nazivu. WebApi se na ostatak sustava spaja preko reference na Projekt Business te Data. Iz projekta Business su nam potrebne refrence na Servise, a iz projekta Data model domene.

U sklopu ovog seminara napravljena je također i mobilna aplikacija. Odlučili smo se za iPhone aplikaciju. Aplikacija je napravljena u Xcode razvojnom okruženju koristeći programski jezik Swift. Korištena su i neka gotova rješenja poput Alamofire, Unbox, SVProgressHUD, MBProgressHUD. Integracija sa ostatkom sustava izvedena je spajanjem aplikacije na Web API. Pomoću Alamofirea bilo je lakše vršiti upite na Web API. Unbox je korišten za serijalizaciju odgovora temeljenih na JSON formatu u Swift klase. Prvi primjer prikazuje GET zahtjev za dohvaćanje 20 setova. Moguća je i paginacija sa parametrom offset.

Alamofire.request("http://mylego.local/api/LSetApi/GetAll?take=20",headers: nil).responseJSON { (response:DataResponse<Any>) in

switch response.result {

case .success:

SVProgressHUD.showSuccess(withStatus: "uspjeh")

….

case .failure(let error):

SVProgressHUD.showError(withStatus: "\(error.localizedDescription)")

}

}

POST zahtjevi su vršeni na sličan način. Pritom je bilo potrebno u polje pohraniti niz parametara koje API prima te takav zahtjev poslati ponovo korištenjem Alamofirea.

let parameters: Parameters = [

"Email": username,

"Password": password

]

Alamofire.request("http://mylego.local/api/UserApi/Login", method: .post, parameters: parameters, encoding: JSONEncoding.default, headers: nil).responseJSON { (response:DataResponse<Any>) in

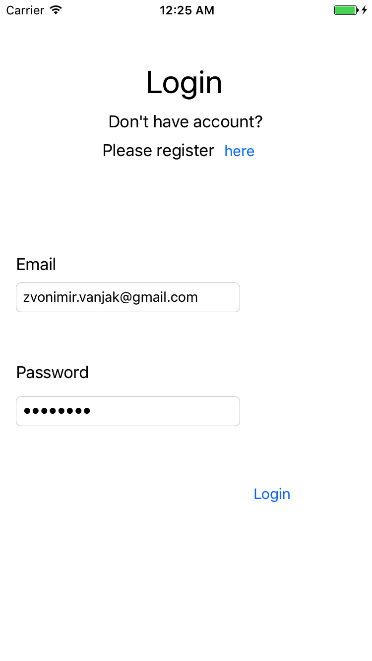
switch(response.result) {

...

}

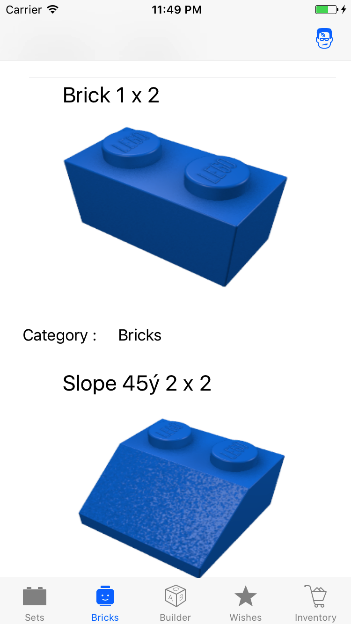
}

Rezultati dobiveni upitima su potom korišteni za punjenje tableViewa ovisno o zahtjevima zadatka. Aplikacija sadrži 8 aktivnih ekrana. Na početku korisnik se mora prijaviti ili registrirati ovisno o tome da li ima korisnički račun za navedenu aplikaciju.



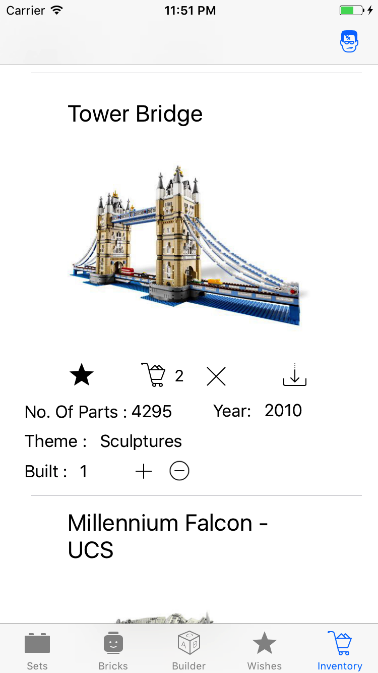
Slika . Slika ekrana za registraciju

Korisnik putem navigacijske trake na dnu pristupa pregledu svih setova, pregledu svih kockica, Builder Assistantu, listi želja ili inventaru. U gornjem desnu kutu aplikacije nalazi se i poveznica za korisnički profil. Pregled svih setova omogućava dodavanje seta u listu želja i inventar, te pristup uputama za slaganje određenog seta. Upute se za korisnike otvaraju u Safari pregledniku. Većinom se sastoje od više knjižica stoga korisnik može odabrati koju treba.



Slika . Pregled kockica

U prozoru inventara korisnik može dodati set u listu želja, pristupiti uputama, te dodavati i brisati setove iz inventara. Korisnik dodavanje i brisanje vrši po jednom komadu seta koji posjeduje. Također omogućeno je i vođenje evidencije izgrađenih setova od ponuđenih. To se vrši atributom “built” koji ima button + i -.



Slika . Inventory

Prozor lista želja nudi analogan način rada. Omogućeno je i brisanje iz liste želja.