**KAUNO TECHNOLOGIJOS UNIVERSITETAS**

**INFORMATIKOS FAKULTETAS**

**OBJEKTINIS PROGRAMŲ PROJEKTAVIMAS (T120B516)**

**Inžinerinis projektas**

Darbą rengė:

Justas Valotka, IFF-5/8

Matas Kulkovas, IFF-5/5

Rytis Pranckūnas, IFF-5/6

Darbą priėmė:

lekt. Andrej Ušaniov

**KAUNAS 2018**

**TURINYS**

[**Projekto aprašymas** 4](#_Toc528197747)

[**Žaidimo reikalavimai** 5](#_Toc528197748)

[**Žaidimo Use Case diagrama** 5](#_Toc528197749)

[**Projekto klasių diagramos** 6](#_Toc528197750)

[**Šablonai** 9](#_Toc528197751)

[**Singleton** 9](#_Toc528197752)

[**Factory** 10](#_Toc528197753)

[**Strategy** 12](#_Toc528197754)

[**Observer** 14](#_Toc528197755)

[**Adapter** 15](#_Toc528197756)

[**Decorator** 16](#_Toc528197757)

[**Command** 17](#_Toc528197758)

[**Facade** 19](#_Toc528197759)

**PAVEIKSLĖLIAI**

[pav 1. "Swords and Sandals" žaidimo langas - projekto orientyras 4](#_Toc528197760)

[pav 2. Use Case diagrama 5](#_Toc528197761)

[pav 3. "Client" paketo klasių diagrama 6](#_Toc528197762)

[pav 4. "Server" paketo klasių diagrama 7](#_Toc528197763)

[pav 5. "Objects" paketo klasių diagrama 8](#_Toc528197764)

[pav 6. Singleton šablono diagrama 9](#_Toc528197765)

[pav 7. Factory šablono diagrama 10](#_Toc528197766)

[pav 8. Strategy šablono diagrama 12](#_Toc528197767)

[pav 9. Observer šablono diagrama 14](#_Toc528197768)

[pav 10. Adapter šablono diagrama 15](#_Toc528197769)

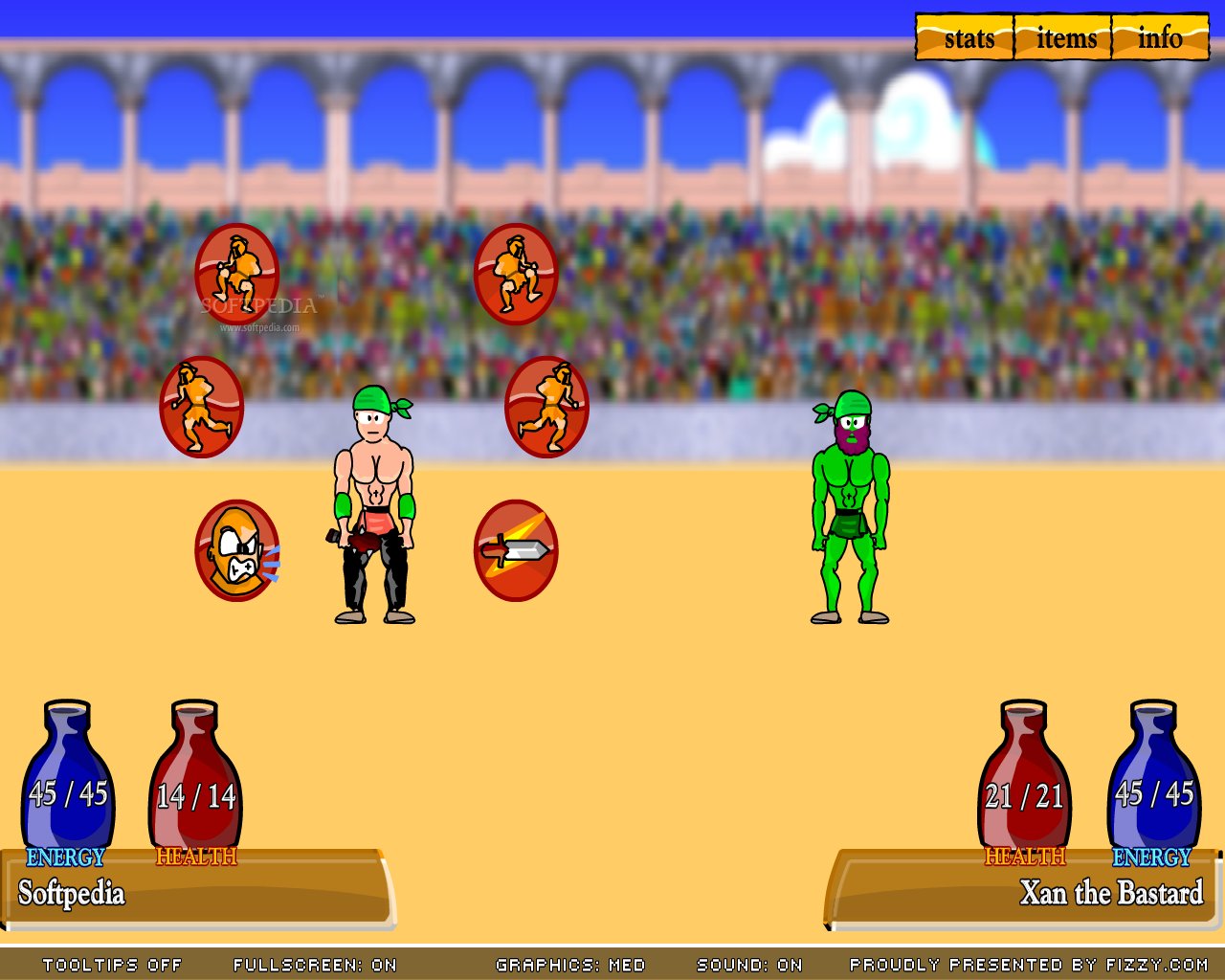
[pav 11. Decorator šablono diagrama 16](#_Toc528197770)

[pav 12. Command šablono diagrama 17](#_Toc528197771)

[pav 13. Facade šablono diagrama 19](#_Toc528197772)

**Projekto aprašymas**

Paprastas, ėjimais pagrįstas 2D kovų žaidimas (2D turn-based fighting game), panašus į tokius žaidimus kaip „Swords and Sandals“, „AdventureQuest“ ar „DragonFable“.

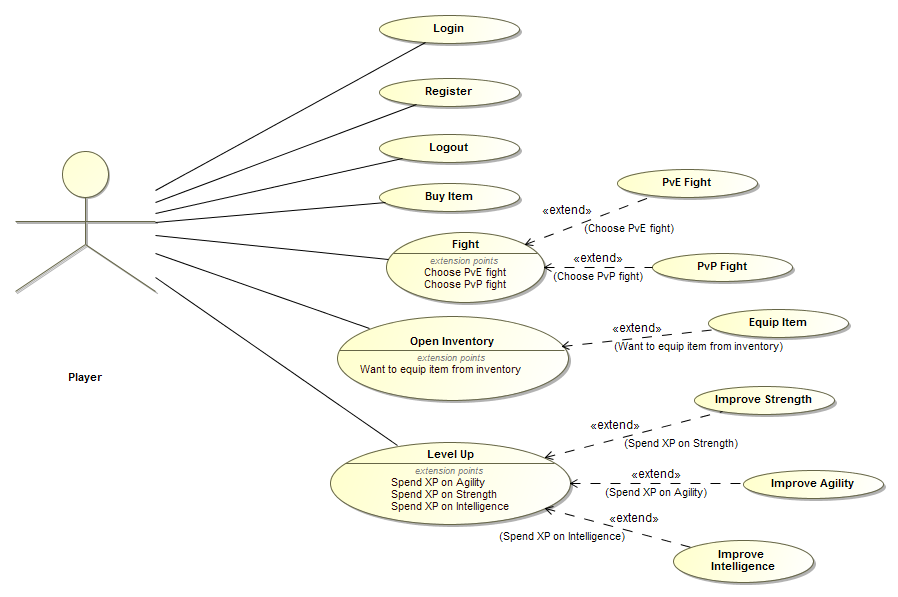


pav 1. "Swords and Sandals" žaidimo langas - projekto orientyras

**Žaidimo reikalavimai**

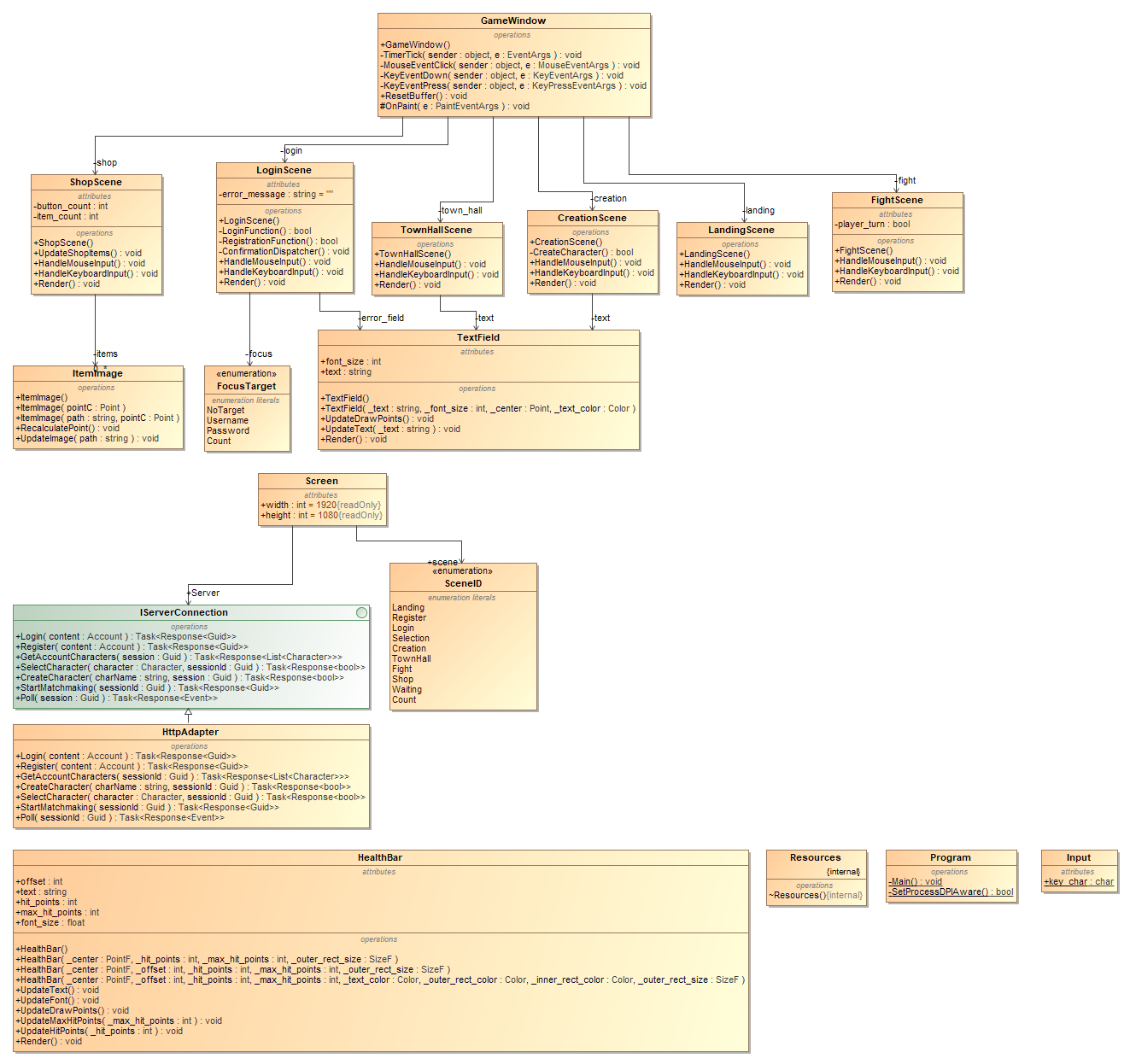
* Galimybė kelti lygį / gauti patirtį (Levels and XP)
* Galimybė gauti auksą ir už jį nusipirkti ginkluotę (Gold)
* Galimybė turėti daug skirtingų ginkluotės vienetų (Backpack/Inventory)
* Kovos žaidime turi vykti ėjimais, paspaudžiant veiksmų mygtukus
* Žaidėjas ir priešai turi turėti atributus (Strength, Agility, Intelligence)
* Žaidėjo ir priešų kovose turimas gyvybių skaičius turi priklausyti nuo Strength atributo
* Žaidėjo ir priešų kovose daroma žala turi priklausyti nuo Agility atributo
* Žaidėjo ir priešų kovose galimybė du kartus iš eilės paeiti turi priklausyti nuo Intelligence atributo
* Ginkluotės atributai turi būti generuojami
* PvE kovose sutinkamų priešų atributai turi būti generuojami atsižvelgiant į žaidėjo lygį
* Žaidėjas turi galėti užsidėti ginkluotę ir taip gerinti savo atributus bei keisti išvaizdą
* Žaidėjas turi galėti kautis su kitais žaidėjais (PvP)

**Žaidimo Use Case diagrama**

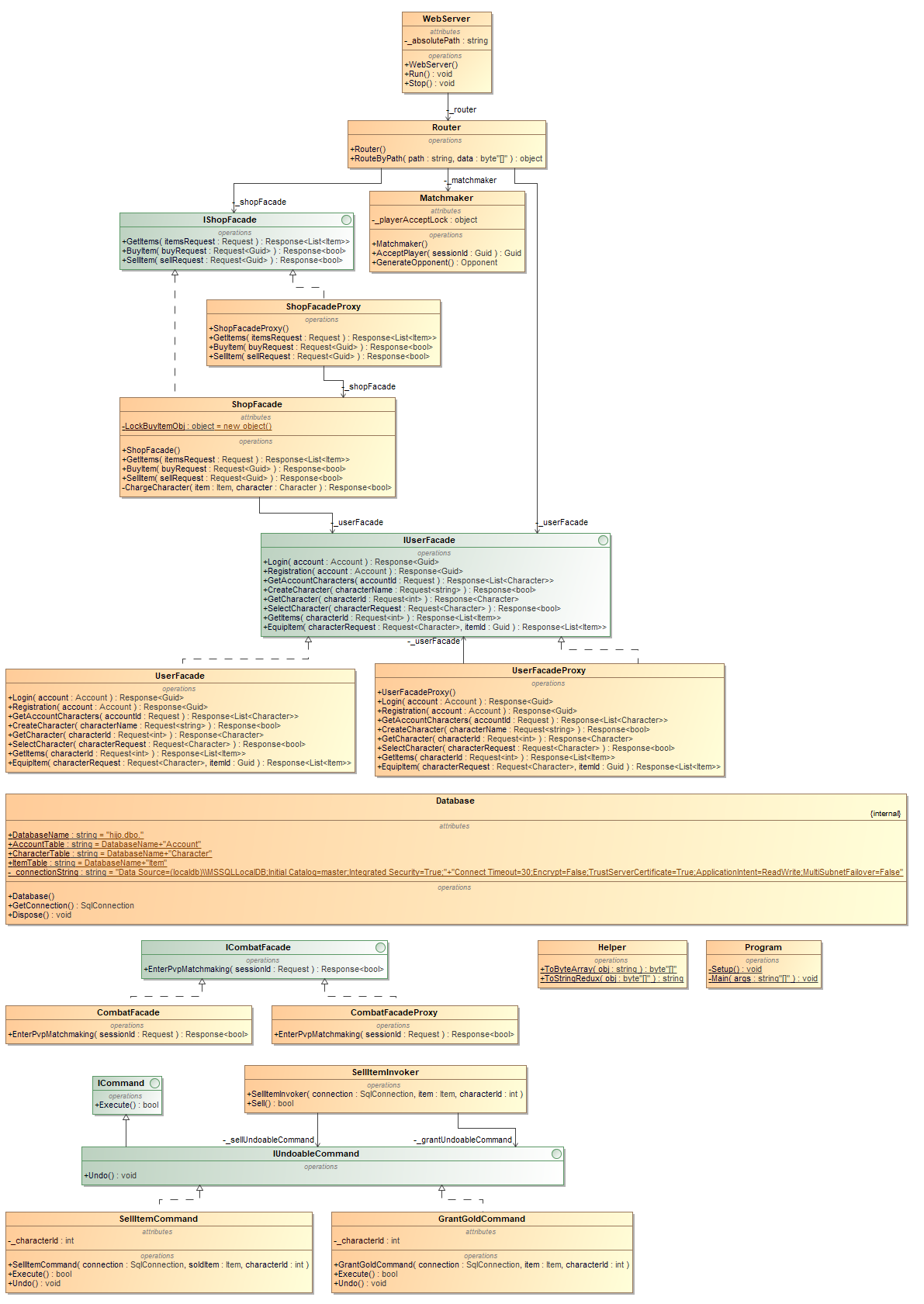


pav 2. Use Case diagrama

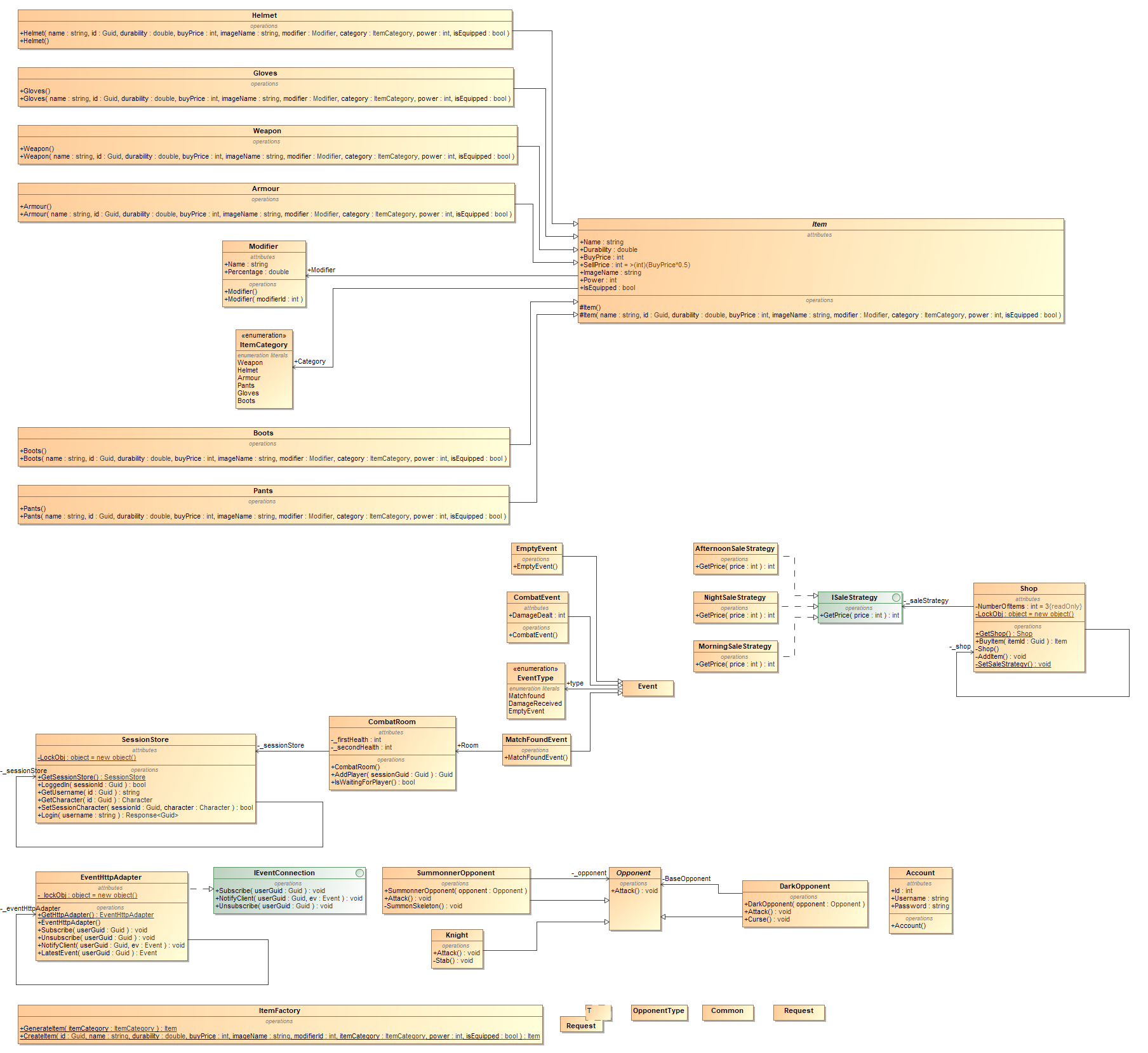
**Projekto klasių diagramos**



pav 3. "Client" paketo klasių diagrama



pav 4. "Server" paketo klasių diagrama

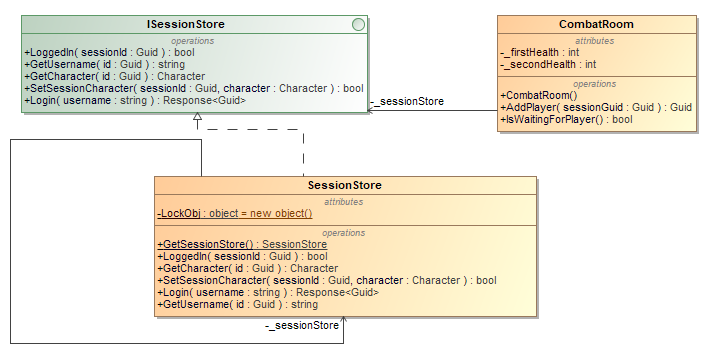


pav 5. "Objects" paketo klasių diagrama

**Šablonai**

**Singleton**

* Klasių diagrama



pav 6. Singleton šablono diagrama

* Esminis kodo fragmentas

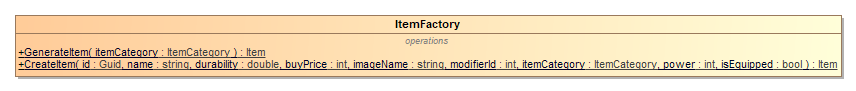
|  |
| --- |
| Objects/SessionStore.cs |
| public class SessionStore  {  private static SessionStore \_sessionStore;  private static readonly Dictionary<Guid, Tuple<string, Character>> Store = new Dictionary<Guid, Tuple<string, Character>>(3);  private static readonly object LockObj = new object();  public static SessionStore GetSessionStore()  {  lock (LockObj)  {  if (\_sessionStore != null)  {  return \_sessionStore;  }  \_sessionStore = new SessionStore();  return \_sessionStore;  }  }  ...  } |

* Naudojimo pagrindimas

Aplikacijoje naudojamas vienas ir tik vienas session store.

**Factory**

* Klasių diagrama



pav 7. Factory šablono diagrama

* Esminis kodo fragmentas

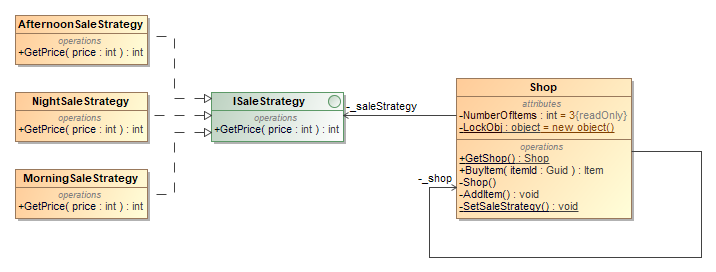
|  |
| --- |
| Objects/ItemFactory.cs |
| public static class ItemFactory  {  public static Item GenerateItem(ItemCategory? itemCategory = null)  {  switch (itemCategory ?? (ItemCategory)Common.Random.Next(Enum.GetNames(typeof(ItemCategory)).Length))  {  case ItemCategory.Armour:  return new Armour();  case ItemCategory.Boots:  return new Boots();  case ItemCategory.Gloves:  return new Gloves();  case ItemCategory.Helmet:  return new Helmet();  case ItemCategory.Pants:  return new Pants();  case ItemCategory.Weapon:  return new Weapon();  default:  throw new ArgumentOutOfRangeException();  }  }  public static Item CreateItem(Guid id, string name, double durability, int buyPrice, string imageName, int modifierId, ItemCategory itemCategory, int power, bool isEquipped)  {  var modifier = new Modifier(modifierId);  switch (itemCategory)  {  case ItemCategory.Armour:  return new Armour(name, id, durability, buyPrice, imageName, modifier, itemCategory, power, isEquipped);  case ItemCategory.Boots:  return new Boots(name, id, durability, buyPrice, imageName, modifier, itemCategory, power, isEquipped);  case ItemCategory.Gloves:  return new Gloves(name, id, durability, buyPrice, imageName, modifier, itemCategory, power, isEquipped);  case ItemCategory.Helmet:  return new Helmet(name, id, durability, buyPrice, imageName, modifier, itemCategory, power, isEquipped);  case ItemCategory.Pants:  return new Pants(name, id, durability, buyPrice, imageName, modifier, itemCategory, power, isEquipped);  case ItemCategory.Weapon:  return new Weapon(name, id, durability, buyPrice, imageName, modifier, itemCategory, power, isEquipped);  default:  throw new ArgumentOutOfRangeException();  }  }  } |

* Naudojimo pagrindimas

Naujų daiktų generavimui.

**Strategy**

* Klasių diagrama



pav 8. Strategy šablono diagrama

* Esminis kodo fragmentas

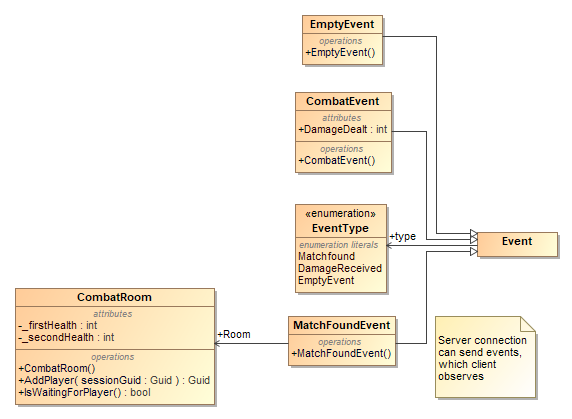
|  |
| --- |
| Server/Shop.cs |
| private static ISaleStrategy \_saleStrategy;  private static readonly object LockObj = new object();  public static Shop GetShop()  {  lock (LockObj)  {  if (\_shop != null)  {  SetSaleStrategy();  return \_shop;  }  \_shop = new Shop();  return \_shop;  }  }  public Item BuyItem(Guid itemId)  {  var item = Items.FirstOrDefault(x => x.Id == itemId);  Items.Remove(item);  AddItem();  return item;  }  private Shop()  {  SetSaleStrategy();  Items = new List<Item>();  for (var i = 0; i < NumberOfItems; i++)  {  AddItem();  }  }  private void AddItem()  {  var item = ItemFactory.GenerateItem();  item.BuyPrice = \_saleStrategy.GetPrice(item.BuyPrice);  Items.Add(item);  } |

* Naudojimo pagrindimas

Serverio lankytojų srautui nukreipti, nepopuliariomis valandomis parduotuvėje atpinga daiktai.

**Observer**

* Klasių diagrama



pav 9. Observer šablono diagrama

* Esminis kodo fragmentas

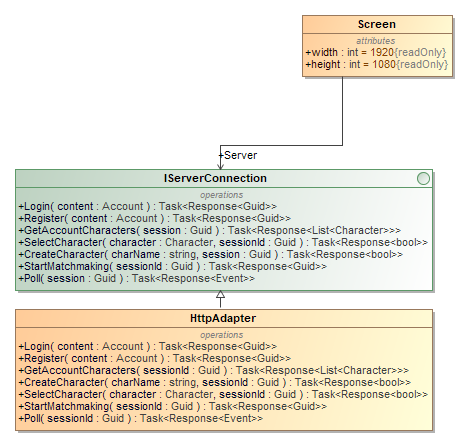
|  |
| --- |
| Server/Matchmaker.cs |
| public Guid AcceptPlayer(Guid sessionId)  {  lock (\_playerAcceptLock)  {  userEvents.Subscribe(sessionId);  var last = combatRooms.LastOrDefault();  if (last == null || !last.IsWaitingForPlayer())  {  var room = new CombatRoom();  combatRooms.Add(room);  return room.AddPlayer(sessionId);  }  var matchFoundEvent = new MatchFoundEvent { Room = last };  userEvents.NotifyClient(last.\_firstGuid, matchFoundEvent);  userEvents.NotifyClient(sessionId, matchFoundEvent);  return last.AddPlayer(sessionId);    }  } |

* Naudojimo pagrindimas

Naudojamas tam, kad galėtumėme naudotojui pranešti apie įvykius.

**Adapter**

* Klasių diagrama



pav 10. Adapter šablono diagrama

* Esminis kodo fragmentas

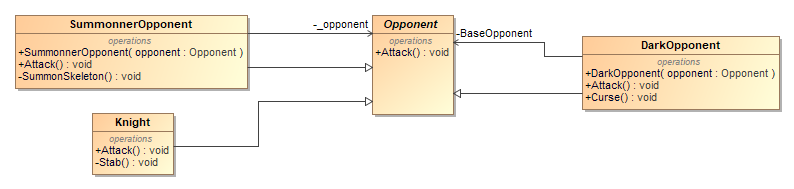
|  |
| --- |
| hijo\_del\_santo/IServerConnection.cs |
| interface IServerConnection  {  Task<Response<Guid>> Login(Account content);  Task<Response<Guid>> Register(Account content);  Task<Response<List<Character>>> GetAccountCharacters(Guid session);  Task<Response<bool>> SelectCharacter(Character character, Guid sessionId);  Task<Response<bool>> CreateCharacter(string charName, Guid session);  Task<Response<Guid>> StartMatchmaking(Guid sessionId);  Task<Response<Event>> Poll(Guid session);  } |

* Naudojimo pagrindimas

Kliento prisijungimas į serverį nepriklauso nuo tinklo protokolo.

**Decorator**

* Klasių diagrama



pav 11. Decorator šablono diagrama

* Esminis kodo fragmentas

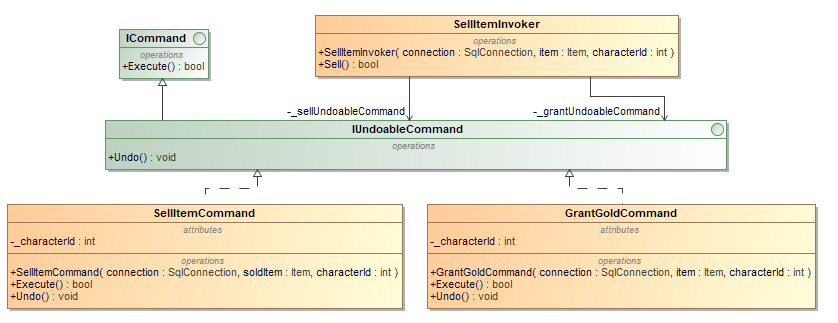
|  |
| --- |
| Server/Matchmaker.cs |
| public Opponent GenerateOpponent()  {  var darkDarkSummonerKnight = new DarkOpponent(new DarkOpponent(new SummonnerOpponent(new Knight())));  return darkDarkSummonerKnight;  } |
| Objects/PveDecorator/DarkOpponent.cs |
| public class DarkOpponent : Opponent  {  private Opponent BaseOpponent { get; set; }  public DarkOpponent(Opponent opponent)  {  BaseOpponent = opponent;  }  public override void Attack()  {  BaseOpponent.Attack();  Curse();  }  public void Curse()  {  }  } |

* Naudojimo pagrindimas

Galima pernaudoti ir kombinuoti priešų ypatingas savybes.

**Command**

* Klasių diagrama



pav 12. Command šablono diagrama

* Esminis kodo fragmentas

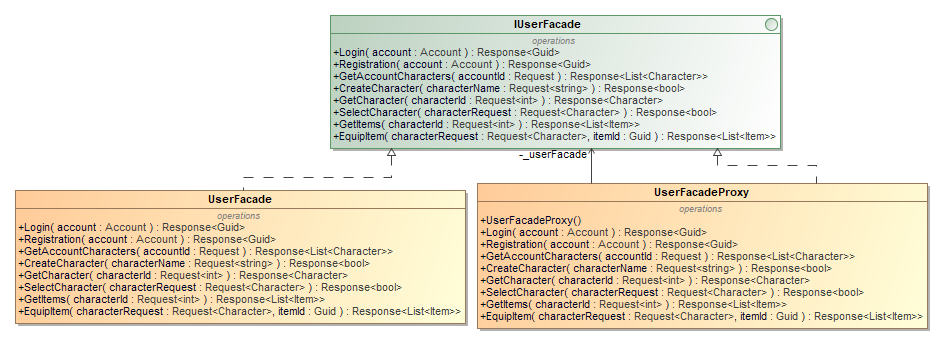
|  |
| --- |
| Server/Commands/SellItemInvoker.cs |
| class SellItemInvoker  {  private readonly IUndoableCommand \_sellUndoableCommand;  private readonly IUndoableCommand \_grantUndoableCommand;  ...  public bool Sell()  {  if (!\_sellUndoableCommand.Execute())  {  return false;  }  if (!\_grantUndoableCommand.Execute())  {  \_sellUndoableCommand.Undo();  return false;  }  return true;  }  } |
| Server/Commands/SellItemCommand.cs |
| public class SellItemCommand : IUndoableCommand  {  private readonly Item \_soldItem;  private readonly int \_characterId;  private readonly SqlConnection \_connection;  ...  public bool Execute()  {  var query = $"Delete from {Database.ItemTable} where CharacterId = '{\_characterId}' and Id = '{\_soldItem.Id}";  using (var command = new SqlCommand(query, \_connection))  {  return command.ExecuteNonQuery() != 0;  }  }  public void Undo()  {  var query =  $"Insert into {Database.ItemTable} (Id, Name, Durability, BuyPrice, ImageName, Modifier, Power, ItemCategory, CharacterId, IsEquipped) " +  $"Values ('{\_soldItem.Id}', '{\_soldItem.Name}', {\_soldItem.Durability}, {\_soldItem.BuyPrice}, '{\_soldItem.ImageName}', " +  $"{\_soldItem.Modifier}, {\_soldItem.Power}, {\_soldItem.Category}, {\_characterId}, {\_soldItem.IsEquipped})";  using (var command = new SqlCommand(query, \_connection))  {  command.ExecuteNonQuery();  }  }  } |

* Naudojimo pagrindimas

Komanda užtikrina, kad pirkimo scenarijuje įvykus klaidai būtų galima atstatyti buvusią būseną.

**Facade**

* Klasių diagrama



pav 13. Facade šablono diagrama

* Esminis kodo fragmentas

|  |
| --- |
| Server/Facades/User/IUserFacade.cs |
| interface IUserFacade  {  Response<Guid> Login(Account account);  Response<Guid> Registration(Account account);  Response<List<Character>> GetAccountCharacters(Request accountId);  Response<bool> CreateCharacter(Request<string> characterName);  Response<Character> GetCharacter(Request<int> characterId);  Response<bool> SelectCharacter(Request<Character> characterRequest);  Response<List<Item>> GetItems(Request<int> characterId);  Response<List<Item>> EquipItem(Request<Character> characterRequest, Guid itemId);  } |

* Naudojimo pagrindimas

Veiksmų su naudotojo paskyra supaprastinimui.