НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КПІ»

Кафедра

Обчислювальної техніки

КУРСОВА РОБОТА
з «ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ»

на тему: «Реалізація робочого місця контакт-менеджера у системі роботи з клієнтами будівельної компанії»

Студента 2 курсу групи ІО-31
напряму підготовки
6.050102 «Комп'ютерна інженерія»

Долинний Олександр Валерійович

Керівник
Болдак Андрій Олександрович
(прізвище та ініціали)

Доцент кафедри ОТ
(посада, вчене звання, науковий ступінь)

Національна шкала_____

Кількість балів: _____

Оцінка: ECTS _____

м. Київ - 2015 рік

					6.050102 «Комп’ютерна інженерія»		
Змн.	Арк.	№ докум.	Підпис	Дата			
Розроб.		Долинний О.В.			Система пошуку та замовлення їжі	Літ.	Арк. 2 / Акрушів 39
Перевір.		Болдак А О					
Реценз.						Кафедра Обчислювальної техніки	
Н. Контр.							
Затверд.		Болдак А О					

РОЗДІЛ 1 ЗАПИТИ ЗАЦІКАВЛЕНИХ ОСІБ

1.1. **Вступ**

У цьому документі описуються запити зацікавлених осіб по відношенню до розроблювальної системи «Будівельна компанія» в якості яких виступають будь-яка фізична чи юридична особа, контакт-менеджер та клієнти цієї компанії.

Гість – майбутній клієнт;

Клієнт – фізична особа, що користується послугами будівельної компанії;

Контакт-менеджер – відповідає за зв’язок компанії з клієнтом;

1.2. **Мета**

Метою документу є визначення основних вимог до функціональності, продуктивності, експлуатаційної придатності, а також визначення бізнес-правил і технологічних обмежень, пред’явлених предмету розробки.

1.3. **Короткий огляд продукту**

Система «Будівельна компанія» надає інтерфейс взаємодії між замовником, персоналом та клієнтами цієї компанії. Замовник має найвищий рівень доступу до організації системи та може контролювати усі внутрішні процеси. Персонал (контакт-менеджер) має доступ до бази даних клієнтів (редагування та оновлення даних клієнта, реєстрація нового клієнта), що дозволяє працівникам напряму спілкуватися з клієнтами. Клієнт має доступ лише до своєї анкети у базі даних та загальної інформації. Гість має доступ лише до загальної інформації.

1.4. **Контекст**

Перелік вимог, зазначених у даному документі є основою технічного завдання для розробки системи «Будівельна компанія».

1.5. **Ділові правила**

1.5.1 **Призначення системи**

Система призначена для зберігання інформації про клієнтів компанії із подальшим використанням на різних рівнях доступу для полегшення організації комунікаційного процесу та економії часу при автоматичному заповненні необхідних документів.

1.5.2 **Політика взаємовідносин**

Система «Будівельна компанія» надає можливість усім працівникам використовувати програмне забезпечення для фіксування результатів переговорів контакт-менеджер – інвестори.

Гість може переглядати список наявних пропозицій та читати форум.

Інвестор має постійний доступ до процесу будівництва за допомогою засобів відеоспостереження, має можливість призначати зустріч з контакт-менеджером, переглядати історію переговорів, а також отримувати повну інформацію про інвестоване майно. За політикою конфіденційності системи інвестор має доступ лише до свого профілю, маючи пару логін-пароль, яку він отримує після успішного підписання договору. Інвестор є повноцінним учасником форуму, де може залишати свої повідомлення, оцінювати якість роботи компанії.

Контакт-менеджер має власний робочий кабінет в системі у якій оброблює заявки та повідомлення від інвесторів.

1.5.3 **Характеристика ділового процесу**

Обліковий запис інвестора у системі створюється після укладання угоди з компанією. Після створення нового облікового запису клієнт повинен перенести дані з договору в профіль. По завершенню процедури реєстрації дані відправляються у базу даних компанії.

1.5.4 **Сценарії**

Сценарій першої зустрічі з контакт-менеджером:

1) Гість контактує з компанією, досягають консенсусу про зустріч у назначений час;

2) Контакт-менеджер веде перемовини щодо вибору нерухомості;

3) Контакт-менеджер укладає договір купівлі-продажу між компанією та гостем;

4) Гість отримує статус інвестора та пару логін-пароль від власного облікового запису;

5) Інвестор авторизується у системі і заповнює усі необхідні дані про договір;

6) Контакт-менеджер перевіряє правильність уведених даних інвестором.

Сценарій призначення зустрічі з контакт-менеджером:

1) Інвестор авторизується в системі.

2) Інвестор заходить у розділ системи «Робота з клієнтами».

3) Інвестор натискає на посилання «Залишити повідомлення для адміністрації».

4) Інвестор заповнює поле з назвою « Залишити повідомлення», де вказує причину зустрічі та та натискає кнопку «Надіслати повідомлення».

5) Контакт-менеджер отримує повідомлення.

6) Контакт-менеджер зв’язується з інвестором та призначає зустріч.

Сценарій залишення повідомлення на форумі:

1) Інвестор авторизується в системі.

2) Інвестор заходить у форум системи.

3) Інвестор знаходить потрібний розділ форуму.

4) Інвестор заповнює поле з назвою « Залишити коментар» та натискає кнопку «Надіслати повідомлення».

1.5.5 **Практичність**

Веб-сайт повинен бути оптимізованим для роботи не тільки із комп'ютера, а також із мобільних пристроїв.

Інтерфейс облікового запису повинен відповідати наступним вимогам:

1) Бути зрозумілим і не допускати двозначного тлумачення;

2) Бути виконаним з урахуванням ергономічних вимог, бути інтуїтивно зрозумілим;

3) Усі кодовані параметри або елементи, та наведені скорочення повинні мати тлумачення або вікно-підказку, що буде з'являтися після наведення курсору на елемент або після натискання спеціальної клавіші;

4) При виконанні електронного цифрового підпису в інтерфейс повинен бути включений результат перевірки цілісності цифрового підпису відповідно з сертифікатом. Відомості про порушення цілісного підпису повинні бути виділені окремо.

1.5.6 **Надійність**

Протягом усього терміну зберігання для облікових записів повинна бути забезпечена їх цілісність, незмінність і достовірність.

Для забезпечення збереження та цілісності використовуватиметься метод резервного копіювання.

Система повинна бути добре захищена від різного роду зловмисних атак із метою заволодіння інформації чи атак типу DDoS. Має використовуватися комплекс технологічних і адміністративних процедур, що перешкоджають

випадковій або навмисній зміні збережених записів. Найкращим рішенням є використання електронного цифрового підпису, що дозволяє у будь-який момент перевірити незмінність збереженого запису в порівнянні з моментом його підписання.

Також система повинна витримувати великі навантаження, обслуговуючи значну кількість користувачів.

Крім цього повинна забезпечуватись конфіденційність персональної інформації. Надання доступу до збережених підписаних записів та персональних даних інвесторів здійснюють у відповідності з правами доступу.

РОЗДІЛ 2

РОЗРОБКА ІНФОРМАЦІЙНОГО ЗАБЕЗПЕЧЕННЯ

2.1. **Загальна схема прецедентів для ролі контакт-менеджера**

Загальна схема прецедентів для ролі контакт-менеджера показує можливі послідовності дій контакт-менеджера. Основним видом діяльності контакт-менеджера є робота контакт-менеджера з клієнтами, тобто перевірка валідності даних інвестора, обмін повідомленнями та оформлення фінансового стану інвесторів. Схема прецедентів представлена на рис. 2.1.

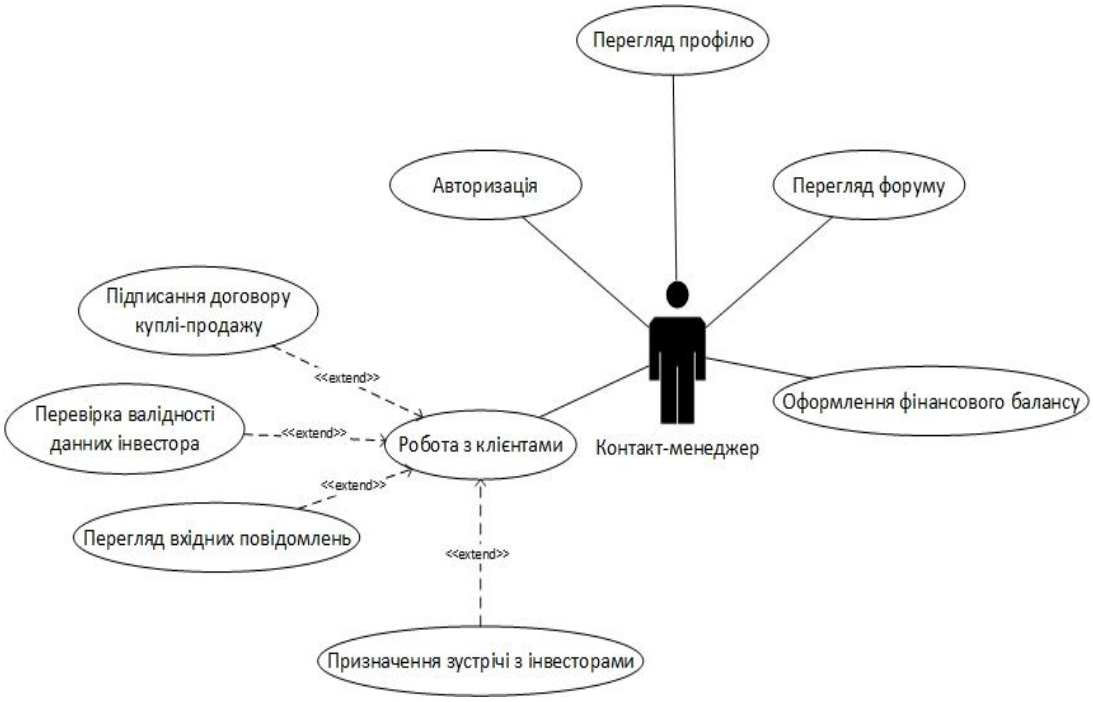


Рис. 2.1 – Загальна схема прецедентів для ролі контакт-менеджера

2.2 **Схема прецедентів для ролі контакт-менеджера**

2.2.1 **Авторизація контакт-менеджера**

ID: UC 001

Назва: Авторизація контакт-менеджера.
Учасники: Контакт-менеджер, система.
Передумови: Контакт-менеджер не має доступа до системи.

Результат: Контакт-менеджер отримує доступ до системи.

Основний сценарій:

1. Контакт-менеджер відправляє запит на авторизацію.
2. Система надає форму із параметрами «Логін» та «Пароль» для авторизації.
3. Контакт-менеджер вводить свої дані (логін/пароль).
4. Контакт-менеджер натискає кнопку «Ввійти в систему».
5. Система перевіряє дані авторизації.
6. Система дає доступ контакт-менеджеру до системи.

Виключні ситуації:

1. Введені персональні дані не вірні (логін/пароль).

2.2.2 Отримання контакт-менеджером зворотнього зв'язку

ID: UC 002

Назва: Отримання контакт-менеджером зворотнього зв'язку.
Учасники: Контакт-менеджер, система.
Передумови: Контакт-менеджер авторизований у системі, контакт-менеджер має необхідність переглянути повідомлення від інвесторів.
Результат: Контакт-менеджер переглядає повідомлення.

Основний сценарій:

1. Контакт-менеджер натискає на кнопку «Переглянути повідомлення».
2. Система переправляє контакт-менеджера на сторінку повідомлень.
3. Контакт-менеджер обирає непрочитане повідомлення і переглядає його.

Виключні ситуації:

1. Контакт-менеджер не має непрочитаних повідомлень.

2.2.3 Оформлення фінансового балансу

ID: UC 003

Назва: Оформлення фінансового балансу.
Учасники: Контакт-менеджер, система.
Передумови: Контакт-менеджер авторизований у системі, інвестор має відредагувати фінансовий баланс певного користувача.

Результат: Система обновлює фінансовий баланс інвестора.

Основний сценарій:

1. Контакт-менеджер натискає на кнопку «Робота з клієнтами».
2. Система відправляє контакт-менеджера на сторінку «Робота з клієнтами».
3. Контакт-менеджер вводить логін інвестора та натискає кнопку «Оновити фінансовий баланс».
4. Система відправляє контакт-менеджера на сторінку із фінансовим балансом обраного інвестора.
5. Контакт-менеджер натискає кнопку «Оновити».
6. Система надає форму контакт-менеджеру для зміни фінансового балансу інвестора.
7. Контакт-менеджер заносить оновлені фінансові дані та дату зміни у форму та натискає на кнопку «Оновити фінансовий баланс».
8. Система оновлює фінансовий баланс інвестора.

Виключні ситуації:

1. Контакт-менеджер неправильно ввів логін інвестора.
2. Контакт-менеджер не заповнив якесь з полів.

2.2.4 **Перевірка валідності введених даних інвестора**

ID: UC 004

Назва: Перевірка валідності введених даних інвестора.
Учасники: Контакт-менеджер, система.
Передумови: Система відправила запит на перевірку валідності введених даних інвестора контакт-менеджеру.
Результат: Інвестор отримує права інвестора у системі.

Основний сценарій:

1. Контакт-менеджер натискає на кнопку «Робота з клієнтами».
2. Система відправляє контакт-менеджера на сторінку «Робота з клієнтами».
3. Контакт-менеджер вводить логін інвестора та натискає кнопку «Перевірити дані інвестора».
4. Система відправляє контакт-менеджера на сторінку із даними обраного інвестора.
5. Контакт-менеджер перевіряє введені дані інвестора.
6. Контакт-менеджер натискає кнопку «Перевірено».
7. Система надає інвестору повні права інвестора.

Виключні ситуації:

1. Введені дані інвестора не валідні.

2.2.5 **Перегляд профілю**

ID: UC 005

Назва: Перегляд профілю.
Учасники: Користувач, система.
Передумови: Користувач авторизований у системі.
Результат: Користувач переглядає свій профіль.

Основний сценарій:

1. Користувач натискає на кнопку “Переглянути профіль”.
2. Система відправляє користувача на сторінку «Мій профіль».
3. Користувач обирає категорію персональних даних користувача та натискає кнопку «Переглянути».
4. Система відправляє користувача на потрібну сторінку.
5. Користувач переглядає обрану інформацію.

2.3 **Діаграма бізнес-сутностей**

Дана діаграма створюється на етапі бізнес моделювання. Вона відображає основні сутності та взаємозв’язки між ними. Діаграма бізнес-сутностей проекту зображена на рис. 2.8.

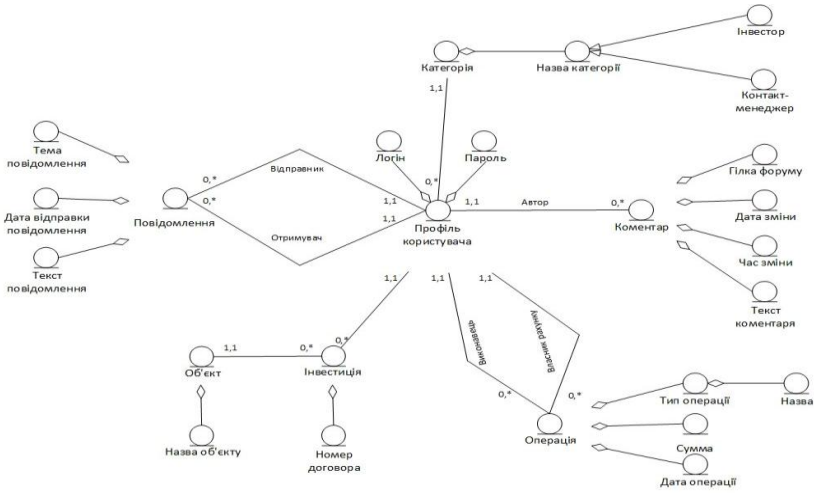


Рис. 2.8 – Діаграма бізнес-сутностей

2.4. Реляційна модель бази даних

Реляційна модель бази даних (рис 2.3) зображує структуру таблиць бази даних, взаємозв'язки між ними та поля кожної з таблиць. Наведена діаграма має багато схожого з діаграмою бізнес-сутностей. Кожній основній бізнес-сутності відповідає таблиця баз даних. Script для створення бази даних наведений у додатку А

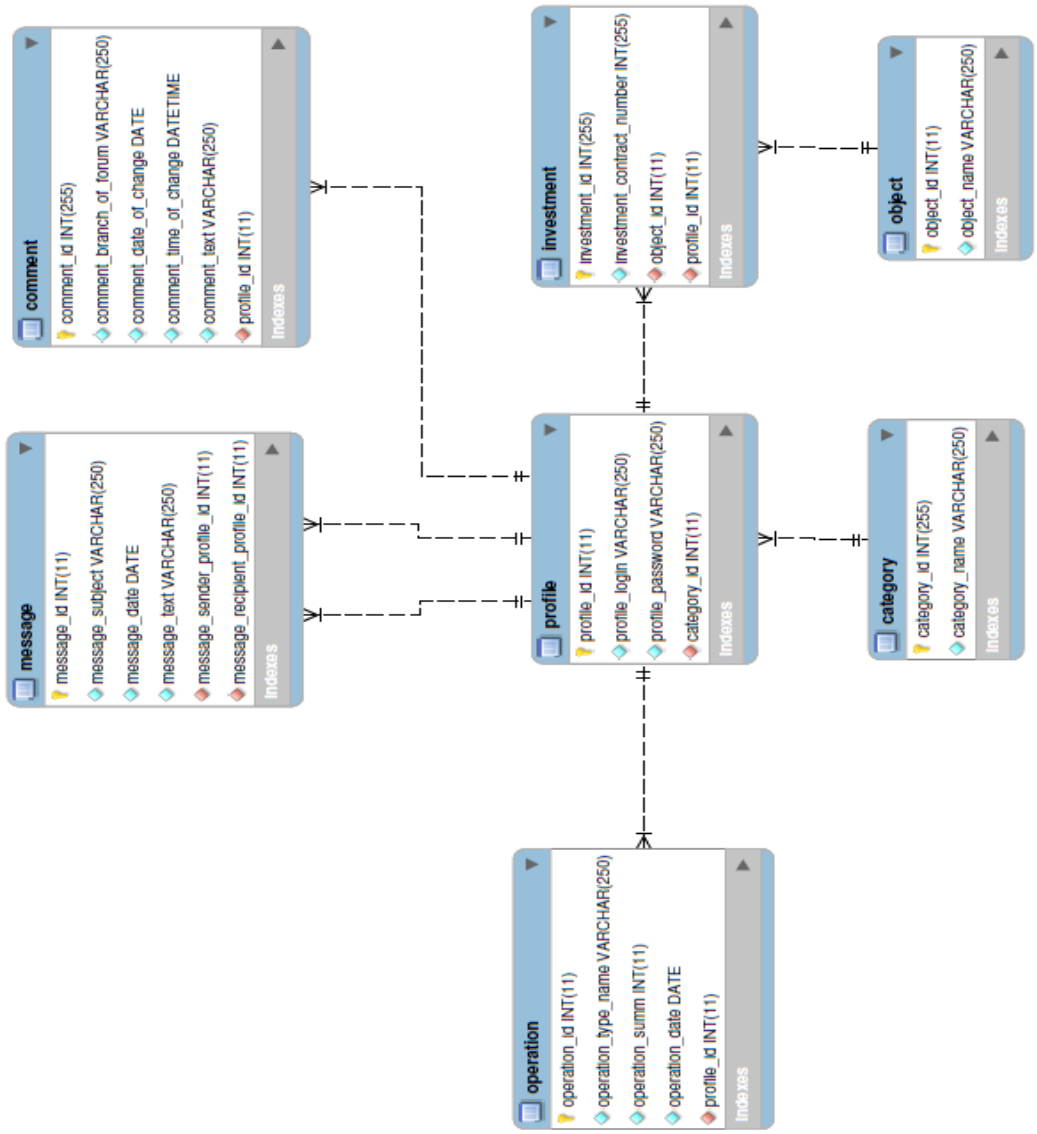


Рис 2.9 – Реляційна модель

РОЗДІЛ 3

РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

3.1. **Реляційно-об’єктне відображення**

Для реляційно-об’єктного відображення в програмі використовується Java Persistence API. Вона надає можливість легко встановити зв’язок з будь-якою базою даних та створити відображення між об’єктно-орієнтованою моделлю та традиційною реляційною моделлю баз даних. На рис. 3.1 зображено діаграму Entity класів. Детальна специфікація (JavaDoc) наведена нижче.

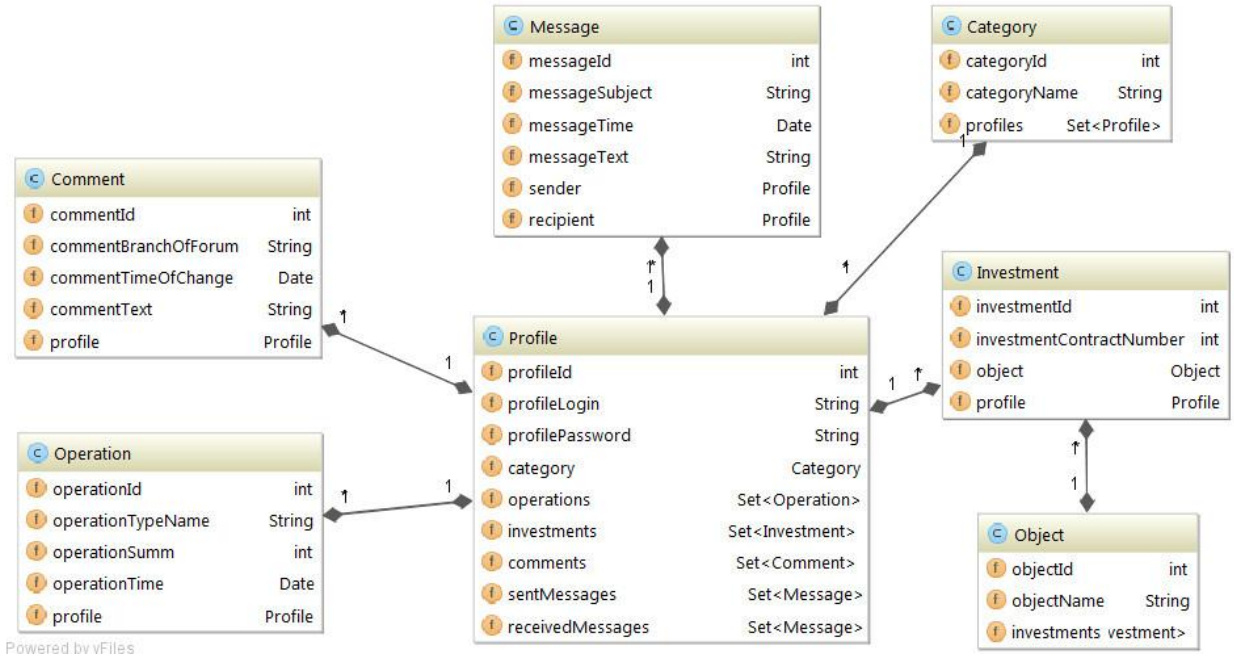


Рис 3.1 – Діаграма entity класів

3.1.1. Клас «Profile»

```
@Entity
public class Profile
extends java.lang.Object
```

Constructor Summary

Constructors

Constructor and Description
`Profile()`

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method and Description
Category	getCategory()
java.util.Set<Comment>	getComments()
java.util.Set<Investment>	getInvestments()
java.util.Set<Operation>	getOperations()
int	getProfileId()
java.lang.String	getProfileLogin()
java.lang.String	getProfilePassword()
java.util.Set<Message>	getReceivedMessages()
java.util.Set<Message>	getSentMessages()
void	setCategory(Category category)
void	setComments(java.util.Set<Comment> comments)
void	setInvestments(java.util.Set<Investment> investments)
void	setOperations(java.util.Set<Operation> operations)
void	setProfileId(int profileId)
void	setProfileLogin(java.lang.String profileLogin)
void	setProfilePassword(java.lang.String profilePassword)
void	setReceivedMessages(java.util.Set<Message> receivedMessages)
void	setSentMessages(java.util.Set<Message> sentMessages)
java.lang.String	toString()

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

3.1.2. Клас «Operation»

```
@Entity
public class Operation
extends java.lang.Object
```

Constructor Summary

Constructors

Constructor and Description
Operation()

Method Summary

All Methods	Instance Methods	Concrete Methods

Modifier and Type	Method and Description
int	getOperationId()
int	getOperationSumm()
java.util.Date	getOperationTime()
java.lang.String	getOperationTypeName()
Profile	getProfile()
void	setOperationId(int operationId)
void	setOperationSumm(int operationSumm)
void	setOperationTime(java.util.Date operationTime)
void	setOperationTypeName(java.lang.String operationTypeName)
void	setProfile(Profile profile)
java.lang.String	toString()

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

3.1.3. Клас «Message»

```
@Entity
public class Message
extends java.lang.Object
```

Constructor Summary

Constructors

Constructor and Description
Message()

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method and Description
java.lang.String	getMessageSubject()
java.lang.String	getMessageText()
java.util.Date	getMessageTime()
Profile	getRecipient()
Profile	getSender()
void	setMessageSubject(java.lang.String messageSubject)
void	setMessageText(java.lang.String messageText)
void	setMessageTime(java.util.Date messageTime)
void	setRecipient(Profile recipient)
void	setSender(Profile sender)
java.lang.String	toString()

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

3.1.4. Клас «Category»

```
@Entity
public class Category
extends java.lang.Object
```

Constructor Summary

Constructors

Constructor and Description
Category()

Method Summary

All Methods	Instance Methods	Concrete Methods

Modifier and Type	Method and Description
int	getCategoryId()
java.lang.String	getCategoryName()
java.util.Set<Profile>	getProfiles()
void	setCategoryId(int categoryId)
void	setCategoryName(java.lang.String categoryName)
void	setProfiles(java.util.Set<Profile> profiles)
java.lang.String	toString()

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

3.1.5. Клас «Comment»

```
@Entity
public class Comment
extends java.lang.Object
```

Constructor Summary

Constructors

Constructor and Description
Comment()

Method Summary

All Methods	Instance Methods	Concrete Methods

Modifier and Type	Method and Description
java.lang.String	getCommentBranchOfForum()
int	getCommentId()
java.lang.String	getCommentText()
java.util.Date	getCommentTimeOfChange()
Profile	getProfile()
void	setCommentBranchOfForum(java.lang.String commentBranchOfForum)
void	setCommentId(int commentId)
void	setCommentText(java.lang.String commentText)
void	setCommentTimeOfChange(java.util.Date commentTimeOfChange)
void	setProfile(Profile profile)
java.lang.String	toString()

3.1.6. Клас «Investment»

```
@Entity
public class Investment
extends java.lang.Object
```

Constructor Summary

Constructors

Constructor and Description
`Investment()`

Method Summary

All Methods | **Instance Methods** | **Concrete Methods**

Modifier and Type	Method and Description
int	getInvestmentContractNumber()
int	getInvestmentId()
Object	getObject()
Profile	getProfile()
void	setInvestmentContractNumber(int investmentContractNumber)
void	setInvestmentId(int investmentId)
void	setObject(Object object)
void	setProfile(Profile profile)
java.lang.String	toString()

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

3.1.7. Клас «Object»

```
@Entity
public class Object
extends java.lang.Object
```

Constructor Summary

Constructors

Constructor and Description
`Object()`

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method and Description
java.util.Set<Investment>	getInvestments()
int	getObjectId()
java.lang.String	getObjectName()
void	setInvestments(java.util.Set<Investment> investments)
void	setObjectId(int objectId)
void	setObjectName(java.lang.String objectName)
java.lang.String	toString()

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

3.2. Специфікація Service класів

Класи, що тут представлені, містять service методи програми. В них закладена вся логіка роботи програми з базою даних. Діаграму цих класів можна побачити на рис. 3.2. Детальна специфікація наведена в додатку В.

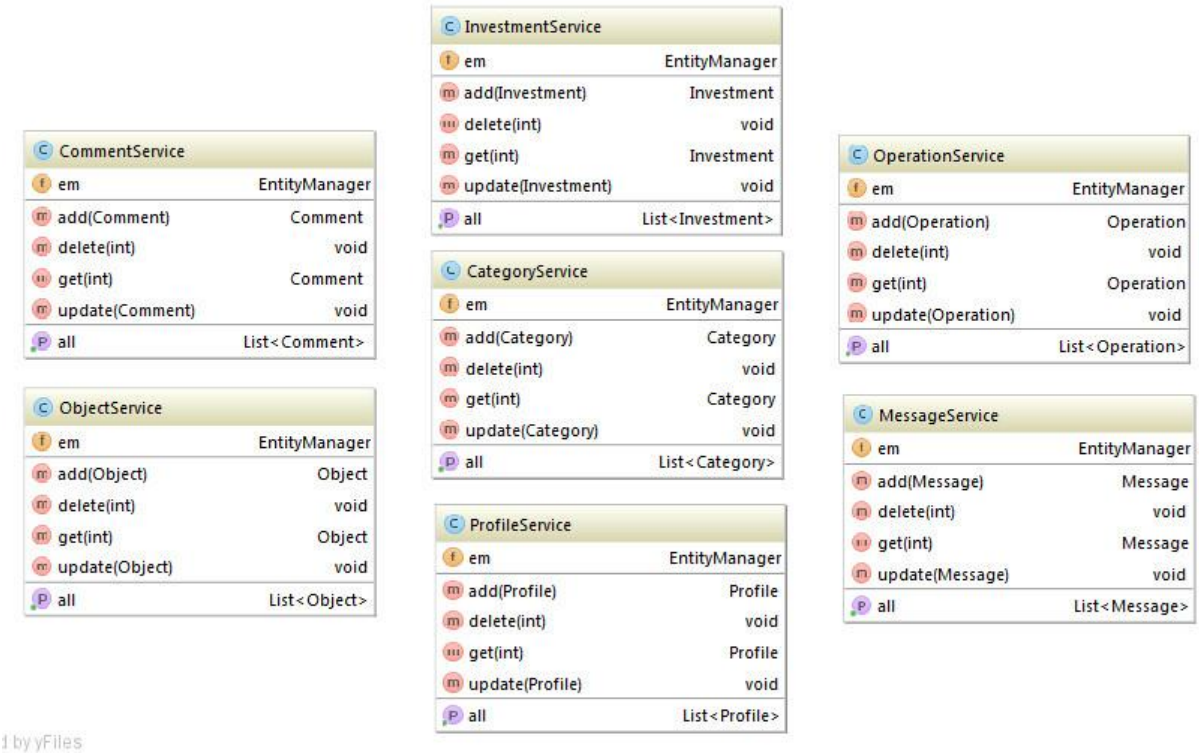


Рис. 3.2 – Діаграма service класів

3.3. **Класи-сервлети та їх специфікація**

Дані класи призначені для створення зв’язку між сервером та клієнтом. Діаграму цих класів можна побачити на рис. 3.3. Детальна специфікація наведена в додатку В.

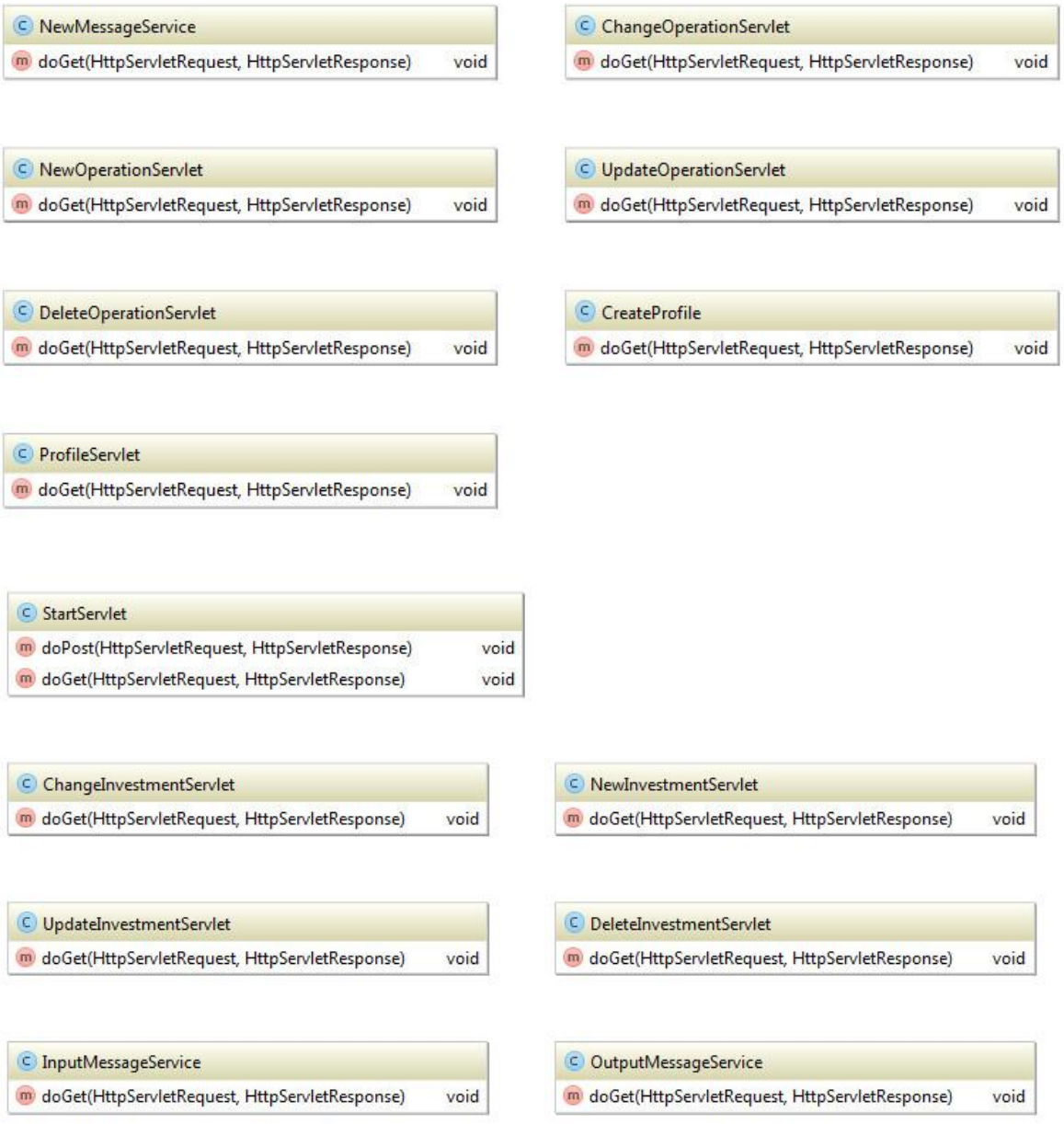
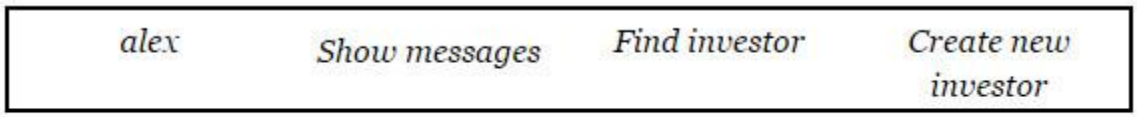


Рис. 3.3 – Діаграма класів-сервлетів

РОЗДІЛ 4

ІЛЮСТРАЦІЯ РОБОТИ ПРОГРАМИ

Для ілюстрації роботи програми в цьому розділі наведено графічні сценарії роботи проекту.



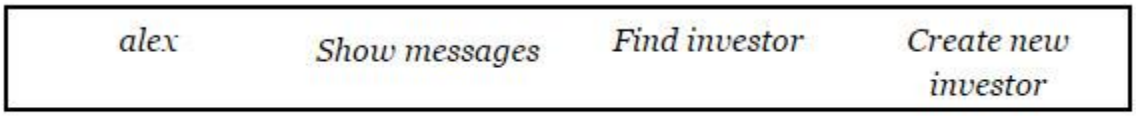
Subject:	ref
Time:	2015-04-29 07:29:33.0
Receiver:	ololosh
Text:	i love you

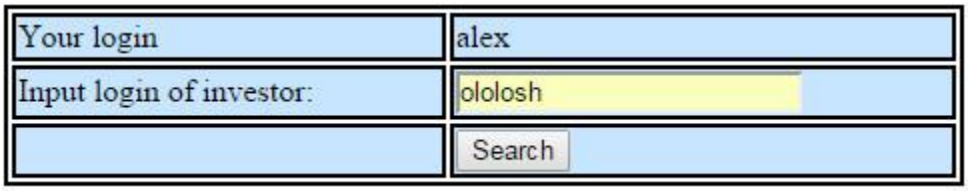
4.1. **Авторизація контакт-менеджера та перехід на профіль**

При заході на сайт без попередньої авторизації, система видає контакт-менеджеру форму для введення логіну та паролю. Контакт-менеджер заповнює поля та натискає кнопку SignIn.

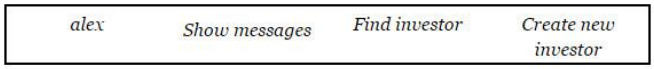
Після авторизації система показує контакт-менеджеру сторінку зі списком дозволених контакт-менеджерові дій. Контакт-менеджер обирає необхідну й натискає на неї(наприклад, FindInvestor).

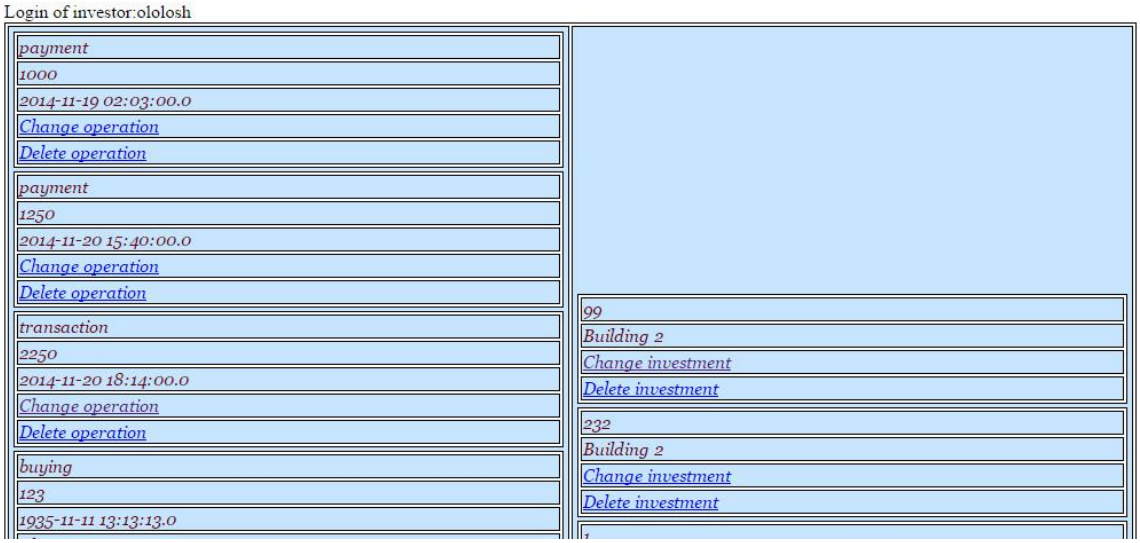
4.2. **Перегляд профілю інвестора**





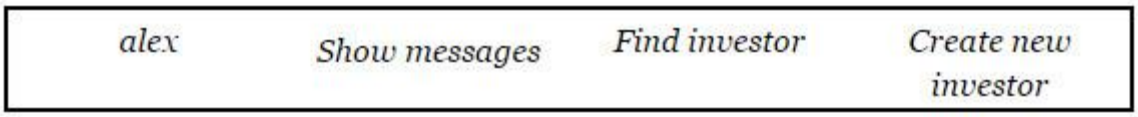
Контакт-менеджер вводить логін інвестора, натискає на кнопку Search та переходить на сторінку профілю інвестора.





4.3. **Відправка повідомлення**

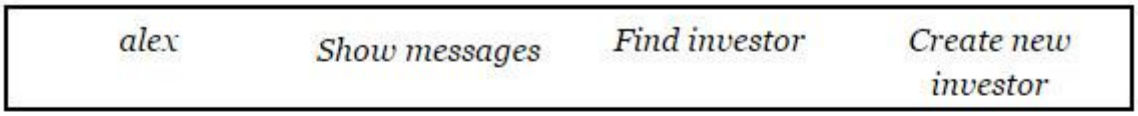
Контакт-менеджер натискає ShowMessages, щоб отримати інформацію про повідомлення.



Your login	alex
Show input messages	
Show output messages	

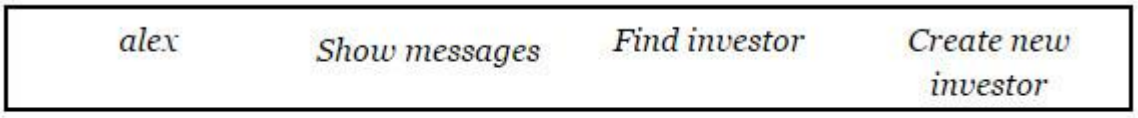
Write new message

Контакт-менеджер натискає WriteNewMessage, щоб написати нове повідомлення.



Your login:	alex
Subject:	ref
Text:	i love you
Recepient:	ololosh
	Send

Контакт-менеджер вводить необхідні параметри, натискає Send, та переправляється на сторінку з відправленими повідомленнями.



Subject:	ref
Time:	2015-04-29 07:29:33.0
Receiver:	ololosh
Text:	i love you

СПИСОК ІНФОРМАЦІЙНИХ ДЖЕРЕЛ

1. Apache Tomcat. – Посилання: http://tomcat.apache.org
2. Hibernate. – Посилання: http://hibernate.org
3. Maven. – Посилання: https://maven.apache.org

ДОДАТОК А

```
CREATE SCHEMA IF NOT EXISTS `building_company` DEFAULT CHARACTER SET utf8 ;
USE ` building_company ` ;


SET FOREIGN_KEY_CHECKS=0;

-- ----------------------------
-- Table structure for category
-- ----------------------------
DROP TABLE IF EXISTS `category`;
CREATE TABLE `category` (
  `category_id` int(255) NOT NULL AUTO_INCREMENT,
  `category_name` varchar(250) NOT NULL,
  PRIMARY KEY (`category_id`),
  KEY `profile_id` (`category_id`)
) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=utf8;

-- ----------------------------
-- Records of category
-- ----------------------------
INSERT INTO `category` VALUES ('1', 'Investor');
INSERT INTO `category` VALUES ('2', 'Contact-manager');

-- ----------------------------
-- Table structure for comment
-- ----------------------------
DROP TABLE IF EXISTS `comment`;
CREATE TABLE `comment` (
  `comment_id` int(255) NOT NULL AUTO_INCREMENT,
  `comment_branch_of_forum` varchar(250) NOT NULL,
  `comment_date_of_change` date NOT NULL,
  `comment_time_of_change` datetime NOT NULL,
  `comment_text` varchar(250) NOT NULL,
  `profile_id` int(11) NOT NULL,
  PRIMARY KEY (`comment_id`),
  KEY `profile_id` (`profile_id`) USING BTREE,
  CONSTRAINT `comment_pp` FOREIGN KEY (`profile_id`) REFERENCES `profile` (`profile_id`) ON DELETE
NO ACTION ON UPDATE NO ACTION
) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=utf8;

-- ----------------------------
-- Records of comment
-- ----------------------------
INSERT INTO `comment` VALUES ('1', 'Fresh news', '2014-11-25', '2014-11-25 21:18:23.000000', 'so interesting',
'1');
INSERT INTO `comment` VALUES ('2', 'FAQ', '2014-11-20', '2014-11-20 21:42:51.000000', 'useful', '2');
INSERT INTO `comment` VALUES ('3', 'Prices', '2014-11-21', '2014-11-21 21:43:05.000000', 'wtf?!', '3');

-- ----------------------------
-- Table structure for investment
-- ----------------------------
DROP TABLE IF EXISTS `investment`;
CREATE TABLE `investment` (
  `investment_id` int(255) NOT NULL AUTO_INCREMENT,
  `investment_contract_number` int(255) NOT NULL,
  `object_id` int(11) NOT NULL,
  `profile_id` int(11) NOT NULL,
  PRIMARY KEY (`investment_id`),
  KEY `object_id` (`object_id`) USING BTREE,
  KEY `investment_pp` (`profile_id`),
  CONSTRAINT `investment_pp` FOREIGN KEY (`profile_id`) REFERENCES `profile` (`profile_id`),
```

```
  CONSTRAINT `investment_oo` FOREIGN KEY (`object_id`) REFERENCES `object` (`object_id`)
) ENGINE=InnoDB AUTO_INCREMENT=6 DEFAULT CHARSET=utf8;

-- ----------------------------
-- Records of investment
-- ----------------------------
INSERT INTO `investment` VALUES ('1', '5123151', '1', '1');
INSERT INTO `investment` VALUES ('2', '5516516', '1', '1');
INSERT INTO `investment` VALUES ('3', '5621533', '2', '2');
INSERT INTO `investment` VALUES ('4', '4566138', '2', '2');
INSERT INTO `investment` VALUES ('5', '8815531', '3', '3');

-- ----------------------------
-- Table structure for message
-- ----------------------------
DROP TABLE IF EXISTS `message`;
CREATE TABLE `message` (
  `message_id` int(11) NOT NULL AUTO_INCREMENT,
  `message_subject` varchar(250) NOT NULL,
  `message_date` date NOT NULL,
  `message_text` varchar(250) NOT NULL,
  `message_sender_profile_id` int(11) NOT NULL,
  `message_recipient_profile_id` int(11) NOT NULL,
  PRIMARY KEY (`message_id`),
  KEY `profile_id` (`message_sender_profile_id`,`message_recipient_profile_id`),
  KEY `message_mrp` (`message_recipient_profile_id`),
  CONSTRAINT `message_mrp` FOREIGN KEY (`message_recipient_profile_id`) REFERENCES `profile` (`profile_id`),
  CONSTRAINT `message_msp` FOREIGN KEY (`message_sender_profile_id`) REFERENCES `profile` (`profile_id`)
) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=utf8;

-- ----------------------------
-- Records of message
-- ----------------------------
INSERT INTO `message` VALUES ('1', 'buying', '2014-11-19', 'want to buy', '1', '3');
INSERT INTO `message` VALUES ('2', 'meeting', '2014-11-21', 'want to meet', '2', '3');
INSERT INTO `message` VALUES ('3', 'askingfor question', '2014-11-20', 'how to arrange a meeting with conatct-manager', '2', '1');


-- ----------------------------
-- Table structure for object
-- ----------------------------
DROP TABLE IF EXISTS `object`;
CREATE TABLE `object` (
  `object_id` int(11) NOT NULL AUTO_INCREMENT,
  `object_name` varchar(250) NOT NULL,
  PRIMARY KEY (`object_id`),
  KEY `investment_id` (`object_id`)
) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=utf8;

-- ----------------------------
-- Records of object
-- ----------------------------
INSERT INTO `object` VALUES ('1', 'new building 1');
INSERT INTO `object` VALUES ('2', 'new building 2 ');
INSERT INTO `object` VALUES ('3', 'new building 3');

-- ----------------------------
-- Table structure for operation
-- ----------------------------
DROP TABLE IF EXISTS `operation`;
```

```
CREATE TABLE `operation` (
  `operation_id` int(11) NOT NULL AUTO_INCREMENT,
  `operation_type_name` varchar(250) NOT NULL,
  `operation_summ` int(11) NOT NULL,
  `operation_date` date NOT NULL,
  `profile_id` int(11) NOT NULL,
  PRIMARY KEY (`operation_id`),
  KEY `operation_pp` (`profile_id`),
  CONSTRAINT `operation_pp` FOREIGN KEY (`profile_id`) REFERENCES `profile` (`profile_id`)
) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=utf8;

-- ----------------------------
-- Records of operation
-- ----------------------------
INSERT INTO `operation` VALUES ('1', 'payment', '1000', '2014-11-19', '1');
INSERT INTO `operation` VALUES ('2', 'payment', '1250', '2014-11-20', '2');
INSERT INTO `operation` VALUES ('3', 'transaction', '2250', '2014-11-20', '3');

-- ----------------------------
-- Table structure for profile
-- ----------------------------
DROP TABLE IF EXISTS `profile`;
CREATE TABLE `profile` (
  `profile_id` int(11) NOT NULL AUTO_INCREMENT,
  `profile_login` varchar(250) NOT NULL,
  `profile_password` varchar(250) NOT NULL,
  `category_id` int(11) NOT NULL,
  PRIMARY KEY (`profile_id`),
  KEY `category_id` (`category_id`),
  CONSTRAINT `profile_cc` FOREIGN KEY (`category_id`) REFERENCES `category` (`category_id`)
) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=utf8;



-- ----------------------------
-- Records of profile
-- ----------------------------
INSERT INTO `profile` VALUES ('1', 'ololosh', '123456', '1');
INSERT INTO `profile` VALUES ('2', 'ololoev', '234567', '1');
INSERT INTO `profile` VALUES ('3', 'Nick23', 'qwerty', '2');
```

ДОДАТОК Б

Класс CategoryService.java

```
package logic.service;

/**
 * @author Dolinniy
 * @version 1.0
 * Клас є сутністю, яка використовується для виконання
операцій створення,
 * читання, оновлення та видалення об'єкту Category
 * у відповідній таблиці бази даних
 */

import logic.table.Category;

import javax.persistence.EntityManager;
import javax.persistence.Persistence;
import javax.persistence.TypedQuery;
import java.util.List;

public class CategoryService {

    public EntityManager em =
Persistence.createEntityManagerFactory("BUILDING_COMPANY").c
reateEntityManager();

    /**
     * Метод додає об'єкт Категорію.
     *
     * @param category Категорія
     */
    public Category add(Category category) {
        em.getTransaction().begin();
        Category categoryFromDB = em.merge(category);
        em.getTransaction().commit();
        return categoryFromDB;
    }

    /**
     * Метод видаляє об'єкт Категорію.
     *
     * @param id ідентифікатор об'єкту Категорія
     */
    public void delete(int id) {
        em.getTransaction().begin();
        em.remove(get(id));
        em.getTransaction().commit();
    }

    /**
     * Метод отримує об'єкт Категорію за ідетифікатором.
     *
     * @param id ідентифікатор об'єкту Категорія
     */
    public Category get(int id) {
        return em.find(Category.class, id);
    }

    /**
     * Метод оновлює об'єкт Категорію.
     *
     * @param category Категорія
     */
    public void update(Category category) {
        em.getTransaction().begin();
        em.merge(category);
        em.getTransaction().commit();
    }

    /**
     * Метод отримує список усіх об'єктів класу Категорію.
     */
    public List<Category> getAll() {
        TypedQuery<Category> namedQuery =
em.createNamedQuery("Category.getAll", Category.class);

        return namedQuery.getResultList();
    }

}
```

Класс ProfileService.java

```
package logic.service;

/**
 * @author Dolinniy
 * @version 1.0
 * Клас є сутністю, яка використовується для виконання
операцій створення,
 * читання, оновлення та видалення об'єкту Profile
 * у відповідній таблиці бази даних
 */

import logic.table.Profile;

import javax.persistence.EntityManager;
import javax.persistence.Persistence;
import javax.persistence.TypedQuery;
import java.util.List;

public class ProfileService {

    public EntityManager em =
Persistence.createEntityManagerFactory("BUILDING_COMPANY").c
reateEntityManager();

    /**
     * Метод додає об'єкт Профіль.
     *
     * @param profile Профіль
     */
    public Profile add(Profile profile) {
        em.getTransaction().begin();
        Profile profileFromDB = em.merge(profile);
        em.getTransaction().commit();
        return profileFromDB;
    }

    /**
     * Метод видаляє об'єкт Профіль.
     *
     * @param id ідентифікатор об'єкту Профіль
     */
    public void delete(int id) {
        em.getTransaction().begin();
        em.remove(get(id));
        em.getTransaction().commit();
    }

    /**
     * Метод отримує об'єкт Профіль за ідентифікатором.
     *
     * @param id ідентифікатор об'єкту Профіль
     */
    public Profile get(int id) {
        return em.find(Profile.class, id);
    }

    /**
     * Метод оновлює об'єкт Профіль.
     *
     * @param profile Профіль
     */
    public void update(Profile profile) {
        em.getTransaction().begin();
        em.merge(profile);
        em.getTransaction().commit();
    }

    /**
     * Метод отримує список усіх об'єктів класу Профіль.
*
     */
    public List<Profile> getAll() {
        TypedQuery<Profile> namedQuery =
em.createNamedQuery("Profile.getAll", Profile.class);
        return namedQuery.getResultList();
    }

}
```

Класс ObjectService.java

```
package logic.service;

import logic.table.Object;

import javax.persistence.EntityManager;
import javax.persistence.Persistence;
import javax.persistence.TypedQuery;
import java.util.List;

/**
 * @author Dolinniy
 * @version 1.0
 *          Клас є сутністю, яка використовується для
виконання операцій створення,
 *          читання, оновлення та видалення об'єкту Object
 *          у відповідній таблиці бази даних
 */
public class ObjectService {
```

```
    public EntityManager em =
Persistence.createEntityManagerFactory("BUILDING_COMPANY").c
reateEntityManager();

    /**
     * Метод додає об'єкт Об'єкт.
     *
     * @param object Об'єкт
     */
    public Object add(Object object) {
        em.getTransaction().begin();
        Object objectFromDB = em.merge(object);
        em.getTransaction().commit();
        return objectFromDB;
    }

    /**
     * Метод видаляє об'єкт Об'єкт.
     *
     * @param id ідентифікатор об'єкту Об'єкт
     */
    public void delete(int id) {
        em.getTransaction().begin();
        em.remove(get(id));
        em.getTransaction().commit();
    }

    /**
     * Метод отримує об'єкт Об'єкт за ідетифікатором.
     *
     * @param id ідентифікатор об'єкту Об'єкт
     */
    public Object get(int id) {
        return em.find(Object.class, id);
    }

    /**
     * Метод оновлює об'єкт Об'єкт.
     *
     * @param investment Інвестиція
     */
    public void update(Object investment) {
        em.getTransaction().begin();
        em.merge(investment);
        em.getTransaction().commit();
    }

    /**
     * Метод отримує список усіх об'єктів класу Об'єкт.
     *
     * @param
     */
    public List<Object> getAll() {
        TypedQuery<Object> namedQuery =
em.createNamedQuery("Object.getAll", Object.class);

        return namedQuery.getResultList();
    }
}
```

Клас MessageService.java

```
package logic.service;

import logic.table.Message;

import javax.persistence.EntityManager;
import javax.persistence.Persistence;
import javax.persistence.TypedQuery;
import java.util.List;

/**
 * @author Dolinniy
 * @version 1.0
 *          Клас є сутністю, яка використовується для
виконання операцій створення,
 *          читання, оновлення та видалення об'єкту Message
 *          у відповідній таблиці бази даних
 */
public class MessageService {


    public EntityManager em =
Persistence.createEntityManagerFactory("BUILDING_COMPANY").c
reateEntityManager();

    /**
     * Метод додає об'єкт Повідомлення.
     *
     * @param message Повідомлення
     */
    public Message add(Message message) {
        em.getTransaction().begin();
        Message messageFromDB = em.merge(message);
        em.getTransaction().commit();
        return messageFromDB;
    }

    /**
     * Метод видаляє об'єкт Повідомлення.
     *
     * @param id ідентифікатор об'єкту Повідомлення
     */
    public void delete(int id) {
        em.getTransaction().begin();
        em.remove(get(id));
        em.getTransaction().commit();
    }

    /**
     * Метод отримує об'єкт Повідомлення за ідетифікатором.
     *
     * @param id ідентифікатор об'єкту Повідомлення
     */
    public Message get(int id) {
        return em.find(Message.class, id);
    }

    /**
     * Метод оновлює об'єкт Повідомлення.
     *
     * @param message Повідомлення
     */
    public void update(Message message) {
        em.getTransaction().begin();
        em.merge(message);
        em.getTransaction().commit();
    }

    /**
     * Метод отримує список усіх об'єктів класу
Повідомлення.
     *
     * @param
     */
    public List<Message> getAll() {
        TypedQuery<Message> namedQuery =
em.createNamedQuery("Message.getAll", Message.class);

        return namedQuery.getResultList();
    }
}
```

Клас InvestmentService.java

```
package logic.service;

import logic.table.Investment;

import javax.persistence.EntityManager;
import javax.persistence.Persistence;
import javax.persistence.TypedQuery;
import java.util.List;

/**
 * @author Dolinniy
 * @version 1.0
 *          Клас є сутністю, яка використовується для
виконання операцій створення,
 *          читання, оновлення та видалення об'єкту
Investment
 *          у відповідній таблиці бази даних
 */
public class InvestmentService {

    public EntityManager em =
Persistence.createEntityManagerFactory("BUILDING_COMPANY").c
reateEntityManager();

    /**
     * Метод додає об'єкт Інвестиція.
     *
     * @param investment Інвестиція
     */
    public Investment add(Investment investment) {
        em.getTransaction().begin();
        Investment investmentFromDB = em.merge(investment);
        em.getTransaction().commit();
        return investmentFromDB;
    }

    /**
     * Метод видаляє об'єкт Інвестиція
     *
     * @param id ідентифікатор об'єкту Інвестиція
     */
    public void delete(int id) {
        em.getTransaction().begin();
        em.remove(get(id));
        em.getTransaction().commit();
    }

    /**
     * Метод отримує об'єкт Інвестиція за ідетифікатором.
     *
     * @param id ідентифікатор об'єкту Інвестиція
     */
    public Investment get(int id) {
        return em.find(Investment.class, id);
    }

    /**
     * Метод оновлює об'єкт Інвестиція.
```

```
     *
     * @param investment Інвестиція
     */
    public void update(Investment investment) {
        em.getTransaction().begin();
        em.merge(investment);
        em.getTransaction().commit();
    }

    /**
     * Метод отримує список усіх об'єктів класу Інвестиція.
     */
    public List<Investment> getAll() {
        TypedQuery<Investment> namedQuery =
em.createNamedQuery("Investment.getAll", Investment.class);

        return namedQuery.getResultList();
    }
}
```

Клас CommentService.java

```
package logic.service;

import logic.table.Comment;

import javax.persistence.EntityManager;
import javax.persistence.Persistence;
import javax.persistence.TypedQuery;
import java.util.List;

/**
 * @author Dolinniy
 * @version 1.0
 *          Клас є сутністю, яка використовується для
виконання операцій створення,
 *          читання, оновлення та видалення об'єкту Comment
 *          у відповідній таблиці бази даних
 */
public class CommentService {

    public EntityManager em =
Persistence.createEntityManagerFactory("BUILDING_COMPANY").c
reateEntityManager();

    /**
     * Метод додає об'єкт Коментар.
     *
     * @param comment Коментар
     */
    public Comment add(Comment comment) {
        em.getTransaction().begin();
        Comment commentFromDB = em.merge(comment);
        em.getTransaction().commit();
        return commentFromDB;
    }

    /**
     * Метод видаляє об'єкт Коментар.
     *
     * @param id ідентифікатор об'єкту Коментар
     */
    public void delete(int id) {
        em.getTransaction().begin();
        em.remove(get(id));
        em.getTransaction().commit();
    }

    /**
     * Метод отримує об'єкт Коментар за ідентифікатором.
     *
     * @param id ідентифікатор об'єкту Коментар
     */
    public Comment get(int id) {
        return em.find(Comment.class, id);
    }

    /**
     * Метод оновлює об'єкт Коментар.
     *
     * @param comment Коментар
     */
    public void update(Comment comment) {
        em.getTransaction().begin();
        em.merge(comment);
        em.getTransaction().commit();
    }

    /**
     * Метод отримує список усіх об'єктів класу Коментар.
     */
    public List<Comment> getAll() {
        TypedQuery<Comment> namedQuery =
em.createNamedQuery("Comment.getAll", Comment.class);

        return namedQuery.getResultList();
    }
}
```

Клас StartServlet.java

```
package logic.servlet;

import logic.service.ProfileService;
import logic.table.Profile;


import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.List;

/**
 * @author Dolinniy
 * @version 1.0
 *          <p>
 *          Клас є сервлетом, який використовується при
авторизації
 *          при роботі з сервером
 */
@WebServlet("/authorization")
public class StartServlet extends HttpServlet {
    /**
     * Метод опрацьовує запит та віддає відповідь.
     */
    @Override
    protected void doPost(HttpServletRequest request,
HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html");
        String login = request.getParameter("login");
        String password = request.getParameter("password");
        HttpSession session = request.getSession(true);
        session.setAttribute("login", login);

        ProfileService profileService = new
ProfileService();
        List<Profile> profileList = profileService.getAll();
        PrintWriter out = response.getWriter();
        boolean flag = false;
        for (Profile p : profileList) {
            if ((p.getProfileLogin().compareTo(login) == 0)
&& (p.getProfilePassword().compareTo(password) == 0) &&
(p.getCategory().getCategoryId() == 2))
                flag = true;
        }

        if (flag == true) {
            session.setAttribute("password", password);
            RequestDispatcher dispatcher =
getServletContext().getRequestDispatcher("/entry.jsp");
            dispatcher.forward(request, response);
        } else {
            RequestDispatcher dispatcher =
getServletContext().getRequestDispatcher("/errorAuthorizatio
n.jsp");
            dispatcher.forward(request, response);
        }
    }

    @Override
    protected void doGet(HttpServletRequest request,
HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html");
        String login = request.getParameter("login");
        String password = request.getParameter("password");
        ProfileService profileService = new
ProfileService();
        List<Profile> profileList = profileService.getAll();
        PrintWriter out = response.getWriter();
        boolean flag = false;
        for (Profile p : profileList) {
            if ((p.getProfileLogin().compareTo(login) == 0)
&& (p.getProfilePassword().compareTo(password) == 0))
                flag = true;
        }
        if (flag == true) {
            RequestDispatcher dispatcher =
getServletContext().getRequestDispatcher("entry.jsp");
            dispatcher.forward(request, response);
        } else {
            {
                RequestDispatcher dispatcher =
getServletContext().getRequestDispatcher("errorAuthorization
.jsp");
                dispatcher.forward(request, response);
            }
        }
    }
}
```

Клас CreateProfile.java

```java
package logic.servlet;

import logic.service.CategoryService;

import logic.service.ProfileService;

import logic.table.Profile;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import java.io.IOException;


/**
 * @author Dolinniy
 * @version 1.0
 *          <p>
 *          Клас є сервлетом, який використовується для
створення об'єкту Profile
 *          при роботі з сервером
 */
@WebServlet("/create_profile")
public class CreateProfile extends HttpServlet {
    /**
     * Метод опрацьовує запит та віддає відповідь.
     */
    @Override
    protected void doGet(HttpServletRequest request,
HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html");

        HttpSession session = request.getSession(true);

        String profileLogin =
request.getParameter("profileLogin");
        String profilePassword =
request.getParameter("profilePassword");

        CategoryService categoryService = new
CategoryService();

        Profile profile = new Profile();
        profile.setProfileLogin(profileLogin);
        profile.setProfilePassword(profilePassword);
        profile.setCategory(categoryService.get(1));

        ProfileService profileService = new
ProfileService();
        profileService.add(profile);

        session.setAttribute("loginInvestor", profileLogin);

        RequestDispatcher dispatcher =
getServletContext().getRequestDispatcher("/successCreating.j
sp");
        dispatcher.forward(request, response);


    }
}
```

Клас ProfileServlet.java

```java
package logic.servlet;


import logic.service.InvestmentService;
import logic.service.OperationService;
import logic.service.ProfileService;
import logic.table.Investment;
import logic.table.Operation;
import logic.table.Profile;


import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.ArrayList;
import java.util.List;

/**
 * @author Dolinniy
 * @version 1.0
 *          <p>
 *          Клас є сервлетом, який використовується для
перегляду переліку об'єктів
 *          інвестицій та операцій інвестора  при роботі з
сервером
 */
@WebServlet("/profile")
public class ProfileServlet extends HttpServlet {
    /**
     * Метод опрацьовує запит та віддає відповідь.
     */
    @Override
    protected void doGet(HttpServletRequest request,
HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html");

        HttpSession session = request.getSession(true);
        String loginProfile =
request.getParameter("loginProfile");
        session.setAttribute("loginInvestor", loginProfile);


        ProfileService profileService = new
ProfileService();
        List<Profile> profileList = profileService.getAll();
        PrintWriter out = response.getWriter();
        int id = 0;
        for (Profile p : profileList) {
            if ((p.getProfileLogin().compareTo(loginProfile)
== 0) && (p.getCategory().getCategoryId() == 1))
                id = p.getProfileId();
        }

        if (id != 0) {

            OperationService operationService = new
OperationService();
            List<Operation> operationList =
operationService.getAll();
            ArrayList<Operation> operationList0 = new
ArrayList<Operation>();
            for (Operation i : operationList) {
                if (i.getProfile().getProfileId() == id) {
                    operationList0.add(i);
                }
            }
            session.setAttribute("operationList",
operationList0);

            InvestmentService investmentService = new
InvestmentService();
            List<Investment> investmentList =
investmentService.getAll();
            ArrayList<Investment> investmentList0 = new
ArrayList<Investment>();
            for (Investment i : investmentList) {
                if (i.getProfile().getProfileId() == id) {
                    investmentList0.add(i);
                }
            }
            session.setAttribute("investmentList",
investmentList0);

            RequestDispatcher dispatcher =
getServletContext().getRequestDispatcher("/profile.jsp");
            dispatcher.forward(request, response);

        } else {

            RequestDispatcher dispatcher =
getServletContext().getRequestDispatcher("/errorParameter.js
p");
            dispatcher.forward(request, response);
        }
    }

}
```

Клас StartServlet.java

```java
package logic.servlet;

import logic.service.ProfileService;
import logic.table.Profile;


import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.List;

/**
 * @author Dolinniy
 * @version 1.0
 *          <p>
 *          Клас є сервлетом, який використовується при
авторизації
 *          при роботі з сервером
 */
@WebServlet("/authorization")
public class StartServlet extends HttpServlet {
    /**
```

```
     * Метод опрацьовує запит та віддає відповідь.
     */
    @Override
    protected void doPost(HttpServletRequest request,
HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html");
        String login = request.getParameter("login");
        String password = request.getParameter("password");
        HttpSession session = request.getSession(true);
        session.setAttribute("login", login);

        ProfileService profileService = new
ProfileService();
        List<Profile> profileList = profileService.getAll();
        PrintWriter out = response.getWriter();
        boolean flag = false;
        for (Profile p : profileList) {
            if ((p.getProfileLogin().compareTo(login) == 0)
&& (p.getProfilePassword().compareTo(password) == 0) &&
(p.getCategory().getCategoryId() == 2))
                flag = true;
        }

        if (flag == true) {
            session.setAttribute("password", password);
            RequestDispatcher dispatcher =
getServletContext().getRequestDispatcher("/entry.jsp");
            dispatcher.forward(request, response);
        } else {
            RequestDispatcher dispatcher =
getServletContext().getRequestDispatcher("/errorAuthorizatio
n.jsp");
            dispatcher.forward(request, response);
        }
    }

    @Override
    protected void doGet(HttpServletRequest request,
HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html");
        String login = request.getParameter("login");
        String password = request.getParameter("password");
        ProfileService profileService = new
ProfileService();
        List<Profile> profileList = profileService.getAll();
        PrintWriter out = response.getWriter();
        boolean flag = false;
        for (Profile p : profileList) {
            if ((p.getProfileLogin().compareTo(login) == 0)
&& (p.getProfilePassword().compareTo(password) == 0))
                flag = true;
        }
        if (flag == true) {
            RequestDispatcher dispatcher =
getServletContext().getRequestDispatcher("entry.jsp");
            dispatcher.forward(request, response);
        } else {
            {
                RequestDispatcher dispatcher =
getServletContext().getRequestDispatcher("errorAuthorization
.jsp");
                dispatcher.forward(request, response);
            }
        }
    }
}
```

Клас NewOperationServlet.java

```
package logic.servlet.operation;

import logic.service.InvestmentService;
import logic.service.OperationService;
import logic.service.ProfileService;
import logic.table.Investment;
import logic.table.Operation;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import java.io.IOException;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;

/**
 * @author Dolinniy
 * @version 1.0
 *          <p>
 *          Клас є сервлетом, який використовується для
створення об'єкту Operation
 *          при роботі з сервером
 */
@WebServlet("/new_operation")
public class NewOperationServlet extends HttpServlet {
    /**
     * Метод опрацьовує запит та віддає відповідь.
     *
     * @param request  запит
     * @param response відповідь
     */
    @Override
    protected void doGet(HttpServletRequest request,
HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html");
        HttpSession session = request.getSession(true);

        String operationName =
request.getParameter("operationName");
        int operationSumm =
Integer.parseInt(request.getParameter("operationSumm"));
        String operationTime1 =
request.getParameter("operationTime");
        int profileId =
Integer.parseInt(request.getParameter("profileId"));

        Date operationTime = null;
        try {
            operationTime = new SimpleDateFormat("yyyy-MM-dd
HH:mm:ss").parse(operationTime1);
        } catch (ParseException e) {
            e.printStackTrace();
        }

        ProfileService profileService = new
ProfileService();
        OperationService operationService = new
OperationService();

        Operation operation = new Operation();

        operation.setOperationTypeName(operationName);
        operation.setOperationTime(operationTime);
        operation.setOperationSumm(operationSumm);
        operation.setProfile(profileService.get(profileId));

        operationService.add(operation);

        List<Operation> operationList =
operationService.getAll();
        ArrayList<Operation> operationList0 = new
ArrayList<Operation>();
        for (Operation i : operationList) {
            if (i.getProfile().getProfileId() == profileId)
{
                operationList0.add(i);
            }
        }
        session.setAttribute("operationList",
operationList0);

        InvestmentService investmentService = new
InvestmentService();
        List<Investment> investmentList =
investmentService.getAll();
        ArrayList<Investment> investmentList0 = new
ArrayList<Investment>();
        for (Investment i : investmentList) {
            if (i.getProfile().getProfileId() == profileId)
{
                investmentList0.add(i);
            }
        }
        session.setAttribute("investmentList",
investmentList0);

        RequestDispatcher dispatcher =
getServletContext().getRequestDispatcher("/profile.jsp");
        dispatcher.forward(request, response);

    }

}
```

Клас DeleteOperationServlet.java

```
package logic.servlet.operation;


import logic.service.InvestmentService;
import logic.service.OperationService;
import logic.table.Investment;
import logic.table.Operation;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
```

```
/**
 * @author Dolinniy
 * @version 1.0
 *          <p>
 *          Клас є сервлетом, який використовується для
видалення об'єкту Operation
 *          при роботі з сервером
 */
@WebServlet("/delete_operation")
public class DeleteOperationServlet extends HttpServlet {
    /**
     * Метод опрацьовує запит та віддає відповідь.
     *
     * @param request  запит
     * @param response відповідь
     */
    @Override
    protected void doGet(HttpServletRequest request,
HttpServletResponse response)
            throws ServletException, IOException {

        HttpSession session = request.getSession(true);

        response.setContentType("text/html");
        int opId =
Integer.parseInt(request.getParameter("opId"));

        OperationService operationService = new
OperationService();
        Operation operation = operationService.get(opId);
        int profileId =
operation.getProfile().getProfileId();
        operationService.delete(opId);

        List<Operation> operationList =
operationService.getAll();
        ArrayList<Operation> operationList0 = new
ArrayList<Operation>();
        for (Operation i : operationList) {
            if (i.getProfile().getProfileId() == profileId)
{
                operationList0.add(i);
            }
        }
        session.setAttribute("operationList",
operationList0);

        InvestmentService investmentService = new
InvestmentService();
        List<Investment> investmentList =
investmentService.getAll();
        ArrayList<Investment> investmentList0 = new
ArrayList<Investment>();
        for (Investment i : investmentList) {
            if (i.getProfile().getProfileId() == profileId)
{
                investmentList0.add(i);
            }
        }
        session.setAttribute("investmentList",
investmentList0);
        RequestDispatcher dispatcher =
getServletContext().getRequestDispatcher("/profile.jsp");
        dispatcher.forward(request, response);

    }
}
```

Клас ChangeOperationServlet.java

```
package logic.servlet.operation;

import logic.service.OperationService;
import logic.table.Operation;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import java.io.IOException;

/**
 * @author Dolinniy
 * @version 1.0
 *          <p>
 *          Клас є сервлетом, який використовується для
зміни об'єкту Operation
 *          при роботі з сервером
 */
@WebServlet("/change_operation")
public class ChangeOperationServlet extends HttpServlet {
    /**
     * Метод опрацьовує запит та віддає відповідь.
     */
    @Override
    protected void doGet(HttpServletRequest request,
HttpServletResponse response)
            throws ServletException, IOException {

        HttpSession session = request.getSession(true);

        response.setContentType("text/html");
        int opId =
Integer.parseInt(request.getParameter("opId"));

        OperationService operationService = new
OperationService();
        Operation operation = operationService.get(opId);

        session.setAttribute("loginProfile",
request.getParameter("loginProfile"));
        session.setAttribute("opId",
String.valueOf(operation.getOperationId()));
        session.setAttribute("opName",
operation.getOperationTypeName());
        session.setAttribute("opSumm",
String.valueOf(operation.getOperationSumm()));
        session.setAttribute("opTime",
String.valueOf(operation.getOperationTime()));
        session.setAttribute("prId",
String.valueOf(operation.getProfile().getProfileId()));
        RequestDispatcher dispatcher =
getServletContext().getRequestDispatcher("/updateOperation.j
sp");
        dispatcher.forward(request, response);
    }
}
```

Клас OutputMessageService.java

```
package logic.servlet.message;

import logic.service.MessageService;
import logic.service.ProfileService;
import logic.table.Message;
import logic.table.Profile;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

/**
 * @author Dolinniy
 * @version 1.0
 *          <p>
 *          Клас є сервлетом, який використовується для
читання вихідних повідомлень
 *          при роботі з сервером
 */
@WebServlet("/output_message")
public class OutputMessageService extends HttpServlet {
    /**
     * Метод опрацьовує запит та віддає відповідь.
     *
     * @param request  запит
     * @param response відповідь
     */
    @Override
    protected void doGet(HttpServletRequest request,
HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html");

        HttpSession session = request.getSession(true);
        String login = (String)
session.getAttribute("login");

        ProfileService profileService = new
ProfileService();
        List<Profile> profileList = profileService.getAll();
        int profileId = 0;
        for (Profile p : profileList) {
            if (p.getProfileLogin().compareTo(login) == 0)
                profileId = p.getProfileId();
        }

        MessageService messageService = new
MessageService();
        List<Message> messageList = messageService.getAll();
        ArrayList<Message> messageList0 = new
ArrayList<Message>();
        for (Message m : messageList) {
            if (m.getSender().getProfileId() == profileId) {
                messageList0.add(m);
            }
        }
        session.setAttribute("messageList", messageList0);

        RequestDispatcher dispatcher =
getServletContext().getRequestDispatcher("/outputMessage.jsp
");
        dispatcher.forward(request, response);

    }
```

```
}
```

Клас NewMessageService.java

```
package logic.servlet.message;

import logic.service.*;
import logic.table.Investment;
import logic.table.Message;
import logic.table.Operation;
import logic.table.Profile;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;

/**
 * @author Dolinniy
 * @version 1.0
 *          <p>
 *          Клас є сервлетом, який використовується для
створення об'єкту Message
 *          при роботі з сервером
 */
@WebServlet("/new_message")
public class NewMessageService extends HttpServlet {
    /**
     * Метод опрацьовує запит та віддає відповідь.
     *
     * @param request  запит
     * @param response відповідь
     */
    @Override
    protected void doGet(HttpServletRequest request,
HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html");

        HttpSession session = request.getSession(true);
        String login = (String)
session.getAttribute("login");

        ProfileService profileService = new
ProfileService();
        List<Profile> profileList = profileService.getAll();
        int senderId = 0;
        for (Profile p : profileList) {
            if (p.getProfileLogin().compareTo(login) == 0)
                senderId = p.getProfileId();
        }

        String messageSubject =
request.getParameter("messageSubject");
        String messageText =
request.getParameter("messageText");
        String messageRecipient =
request.getParameter("messageRecipient");

        int recipientId = 0;
        for (Profile p : profileList) {
            if
(p.getProfileLogin().compareTo(messageRecipient) == 0)
                recipientId = p.getProfileId();
        }

        Date data = new Date();

        Message message = new Message();
        message.setMessageSubject(messageSubject);
        message.setMessageTime(data);
        message.setMessageText(messageText);
        message.setSender(profileService.get(senderId));

message.setRecipient(profileService.get(recipientId));

        MessageService messageService = new
MessageService();
        messageService.add(message);

        RequestDispatcher dispatcher =
getServletContext().getRequestDispatcher("/output_message");
        dispatcher.forward(request, response);

    }

}
```

Клас InputMessageService.java

```
package logic.servlet.message;


import logic.service.MessageService;
import logic.service.ProfileService;
import logic.table.Message;
import logic.table.Profile;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

/**
 * @author Dolinniy
 * @version 1.0
 *          <p>
 *          Клас є сервлетом, який використовується для
читання вхідних повідомлень
 *          при роботі з сервером
 */
@WebServlet("/input_message")
public class InputMessageService extends HttpServlet {
    /**
     * Метод опрацьовує запит та віддає відповідь.
     *
     * @param request  запит
     * @param response відповідь
     */
    @Override
    protected void doGet(HttpServletRequest request,
HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html");

        HttpSession session = request.getSession(true);
        String login = (String)
session.getAttribute("login");

        ProfileService profileService = new
ProfileService();
        List<Profile> profileList = profileService.getAll();
        int profileId = 0;
        for (Profile p : profileList) {
            if (p.getProfileLogin().compareTo(login) == 0)
                profileId = p.getProfileId();
        }

        MessageService messageService = new
MessageService();
        List<Message> messageList = messageService.getAll();
        ArrayList<Message> messageList0 = new
ArrayList<Message>();
        for (Message m : messageList) {
            if (m.getRecipient().getProfileId() ==
profileId) {
                messageList0.add(m);
            }
        }
        session.setAttribute("messageList", messageList0);

        RequestDispatcher dispatcher =
getServletContext().getRequestDispatcher("/inputMessage.jsp"
);
        dispatcher.forward(request, response);

    }

}
```

Клас UpdateInvestmentServlet.java

```
package logic.servlet.investment;

import logic.service.InvestmentService;
import logic.service.ObjectService;
import logic.service.OperationService;
import logic.service.ProfileService;
import logic.table.Investment;
import logic.table.Operation;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

/**
 * @author Dolinniy
 * @version 1.0
 *
```

```
 * Клас є сервлетом, який використовується для внесення
зміни об'єкту Investment
 * при роботі з сервером
 */
@WebServlet("/update_investment")
public class UpdateInvestmentServlet extends HttpServlet {
    /**
     * Метод опрацьовує запит та віддає відповідь.
     * @param request запит
     * @param response відповідь
     */
    @Override
    protected void doGet(HttpServletRequest request,
HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html");
        HttpSession session = request.getSession(true);
        String loginProfile =
request.getParameter("loginProfile");
        int invId =
Integer.parseInt(request.getParameter("investmentId"));
        int numberContract =
Integer.parseInt(request.getParameter("numberContract"));
        int objectId =
Integer.parseInt(request.getParameter("objectId"));
        int profileId =
Integer.parseInt(request.getParameter("profileId"));

        ProfileService profileService = new
ProfileService();
        ObjectService objectService = new ObjectService();
        InvestmentService investmentService = new
InvestmentService();

        Investment investment = new Investment();
        investment.setInvestmentId(invId);

investment.setInvestmentContractNumber(numberContract);
        investment.setObject(objectService.get(objectId));

investment.setProfile(profileService.get(profileId));

        investmentService.update(investment);

        OperationService operationService = new
OperationService();
        List<Operation> operationList =
operationService.getAll();
        ArrayList<Operation> operationList0 = new
ArrayList<Operation>();
        for (Operation i : operationList) {
            if (i.getProfile().getProfileId() == profileId)
{
                operationList0.add(i);
            }
        }
        session.setAttribute("operationList",
operationList0);

        List<Investment> investmentList =
investmentService.getAll();
        ArrayList<Investment> investmentList0 = new
ArrayList<Investment>();
        for (Investment i : investmentList) {
            if (i.getProfile().getProfileId() == profileId)
{
                investmentList0.add(i);
            }
        }
        session.setAttribute("investmentList",
investmentList0);

        RequestDispatcher dispatcher =
getServletContext().getRequestDispatcher("/profile.jsp");
        dispatcher.forward(request, response);

    }

}
```

Клас NewInvestmentServlet.java

```
package logic.servlet.investment;

import logic.service.InvestmentService;
import logic.service.ObjectService;
import logic.service.OperationService;
import logic.service.ProfileService;
import logic.table.Investment;
import logic.table.Operation;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
```

```
/**
 * @author Dolinniy
 * @version 1.0
 *
 * Клас є сервлетом, який використовується для створення
об'єкту Investment
 * при роботі з сервером
 */
@WebServlet("/new_investment")
public class NewInvestmentServlet extends HttpServlet {
    /**
     * Метод опрацьовує запит та віддає відповідь.
     * @param request запит
     * @param response відповідь
     */
    @Override
    protected void doGet(HttpServletRequest request,
HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html");
        HttpSession session = request.getSession(true);

        int numberContract =
Integer.parseInt(request.getParameter("numberContract"));
        int objectId =
Integer.parseInt(request.getParameter("objectId"));
        int profileId =
Integer.parseInt(request.getParameter("profileId"));


        ProfileService profileService = new
ProfileService();
        ObjectService objectService = new ObjectService();
        InvestmentService investmentService = new
InvestmentService();

        Investment investment = new Investment();


investment.setInvestmentContractNumber(numberContract);
        investment.setObject(objectService.get(objectId));

investment.setProfile(profileService.get(profileId));

        investmentService.add(investment);

        OperationService operationService = new
OperationService();
        List<Operation> operationList =
operationService.getAll();
        ArrayList<Operation> operationList0 = new
ArrayList<Operation>();
        for (Operation i : operationList) {
            if (i.getProfile().getProfileId() == profileId)
{
                operationList0.add(i);
            }
        }
        session.setAttribute("operationList",
operationList0);

        List<Investment> investmentList =
investmentService.getAll();
        ArrayList<Investment> investmentList0 = new
ArrayList<Investment>();
        for (Investment i : investmentList) {
            if (i.getProfile().getProfileId() == profileId)
{
                investmentList0.add(i);
            }
        }
        session.setAttribute("investmentList",
investmentList0);

        RequestDispatcher dispatcher =
getServletContext().getRequestDispatcher("/profile.jsp");
        dispatcher.forward(request, response);

    }

}
```

Клас DeleteInvestmentServlet.java

```
package logic.servlet.investment;

import logic.service.InvestmentService;
import logic.service.OperationService;
import logic.table.Investment;
import logic.table.Operation;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
```

```java
/**
 * @author Dolinniy
 * @version 1.0
 *
 * Клас є сервлетом, який використовується для видалення
об'єкту Investment
 * при роботі з сервером
 */
@WebServlet("/delete_investment")
public class DeleteInvestmentServlet extends HttpServlet {
    /**
     * Метод опрацьовує запит та віддає відповідь.
     * @param request запит
     * @param response відповідь
     */
    @Override
    protected void doGet(HttpServletRequest request,
HttpServletResponse response)
            throws ServletException, IOException {

        HttpSession session = request.getSession(true);

        response.setContentType("text/html");
        int invId =
Integer.parseInt(request.getParameter("invId"));

        InvestmentService investmentService = new
InvestmentService();
        Investment investment =
investmentService.get(invId);
        int profileId =
investment.getProfile().getProfileId();
        investmentService.delete(invId);

        OperationService operationService = new
OperationService();
        List<Operation> operationList =
operationService.getAll();
        ArrayList<Operation> operationList0 = new
ArrayList<Operation>();
        for (Operation i : operationList) {
            if (i.getProfile().getProfileId() == profileId)
{
                operationList0.add(i);
            }
        }
        session.setAttribute("operationList",
operationList0);

        List<Investment> investmentList =
investmentService.getAll();
        ArrayList<Investment> investmentList0 = new
ArrayList<Investment>();
        for (Investment i : investmentList) {
            if (i.getProfile().getProfileId() == profileId)
{
                investmentList0.add(i);
            }
        }
        session.setAttribute("investmentList",
investmentList0);
        RequestDispatcher dispatcher =
getServletContext().getRequestDispatcher("/profile.jsp");
        dispatcher.forward(request, response);

    }
}
```

Клас Profile.java

```java
package logic.table;

import javax.persistence.*;
import java.util.Set;

/**
 * @author Dolinniy
 * @version 1.0
 *          <p>
 *          Клас є сутністю, яка використовується для
моделювання об'єкту Profile
 *          у відповідній таблиці бази даних
 */
@Entity
@Table(name = "profile")
@NamedQuery(name = "Profile.getAll", query = "SELECT c from
Profile c")
public class Profile {

    @Id
    @Column(name = "profile_id")
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int profileId;
    @Column(name = "profile_login")
    private String profileLogin;
    @Column(name = "profile_password")
    private String profilePassword;

    @ManyToOne(fetch = FetchType.EAGER, cascade =
{CascadeType.MERGE, CascadeType.PERSIST,
CascadeType.REFRESH}, optional = false)
    @JoinColumn(name = "category_id", nullable = false)
    private Category category;

    @OneToMany(fetch = FetchType.EAGER, mappedBy =
"profile")
    private Set<Operation> operations;

    @OneToMany(fetch = FetchType.EAGER, mappedBy =
"profile")
    private Set<Investment> investments;

    @OneToMany(fetch = FetchType.EAGER, mappedBy =
"profile")
    private Set<Comment> comments;

    @OneToMany(fetch = FetchType.EAGER, mappedBy = "sender")
    private Set<Message> sentMessages;

    @OneToMany(fetch = FetchType.EAGER, mappedBy =
"recipient")
    private Set<Message> receivedMessages;

    public Set<Investment> getInvestments() {
        return investments;
    }

    public void setInvestments(Set<Investment> investments)
{
        this.investments = investments;
    }

    public int getProfileId() {
        return profileId;
    }

    public void setProfileId(int profileId) {
        this.profileId = profileId;
    }

    public String getProfileLogin() {
        return profileLogin;
    }

    public void setProfileLogin(String profileLogin) {
        this.profileLogin = profileLogin;
    }

    public String getProfilePassword() {
        return profilePassword;
    }

    public void setProfilePassword(String profilePassword) {
        this.profilePassword = profilePassword;
    }

    public Category getCategory() {
        return category;
    }

    public void setCategory(Category category) {
        this.category = category;
    }

    public Set<Operation> getOperations() {
        return operations;
    }

    public void setOperations(Set<Operation> operations) {
        this.operations = operations;
    }

    public Set<Comment> getComments() {
        return comments;
    }

    public void setComments(Set<Comment> comments) {
        this.comments = comments;
    }

    public Set<Message> getSentMessages() {
        return sentMessages;
    }

    public void setSentMessages(Set<Message> sentMessages) {
        this.sentMessages = sentMessages;
    }

    public Set<Message> getReceivedMessages() {
        return receivedMessages;
    }

    public void setReceivedMessages(Set<Message>
receivedMessages) {
        this.receivedMessages = receivedMessages;
    }

    @Override
    public String toString() {
        return "Profile{" +
                "profile_id='" + profileId + '\'' +
                ", profile_login=" + profileLogin +
                ", profile_password=" + profilePassword +
                ", category_id=" + category.getCategoryId()
+
                '}';
    }
}
```

Клас Operation.java

```
package logic.table;

import javax.persistence.*;
import java.util.Date;


/**
 * @author Dolinniy
 * @version 1.0
 *          <p>
 *          Клас є сутністю, яка використовується для
моделювання об'єкту Operation
 *          у відповідній таблиці бази даних
 */
@Entity
@Table(name = "operation")
@NamedQuery(name = "Operation.getAll", query = "SELECT c
from Operation c")
public class Operation {
    @Id
    @Column(name = "operation_id")
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int operationId;
    @Column(name = "operation_type_name")
    private String operationTypeName;
    @Column(name = "operation_summ")

    private int operationSumm;
    @Column(name = "operation_time")
    private java.util.Date operationTime;


    @ManyToOne(fetch = FetchType.EAGER, cascade =
{CascadeType.MERGE, CascadeType.PERSIST,
CascadeType.REFRESH}, optional = false)
    @JoinColumn(name = "profile_id", nullable = false)
    private Profile profile;

    public Profile getProfile() {
        return profile;
    }

    public void setProfile(Profile profile) {
        this.profile = profile;
    }

    public int getOperationId() {
        return operationId;
    }

    public void setOperationId(int operationId) {
        this.operationId = operationId;
    }

    public String getOperationTypeName() {
        return operationTypeName;
    }

    public void setOperationTypeName(String
operationTypeName) {
        this.operationTypeName = operationTypeName;
    }

    public int getOperationSumm() {
        return operationSumm;
    }

    public void setOperationSumm(int operationSumm) {
        this.operationSumm = operationSumm;
    }

    public Date getOperationTime() {
        return operationTime;
    }

    public void setOperationTime(Date operationTime) {
        this.operationTime = operationTime;
    }

    @Override
    public String toString() {
        return "Operation{" +
                "operation_id='" + operationId + '\'' +
                "operation_type_name='" + operationTypeName
+ '\'' +
                ", operation_summ=" + operationSumm +
                ", operation_date=" + operationTime +
                ", profile_id=" + profile.getProfileId() +
                '}';
    }


}
```

Клас Object.java

```
package logic.table;

import javax.persistence.*;
import java.util.Set;

/**
 * @author Dolinniy
 * @version 1.0
 *          <p>
 *          Клас є сутністю, яка використовується для
моделювання об'єкту Object
 *          у відповідній таблиці бази даних
 */
@Entity
@Table(name = "object")
@NamedQuery(name = "Object.getAll", query = "SELECT c from
Object c")
public class Object {
    @Id
    @Column(name = "object_id")
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int objectId;

    @Column(name = "object_name")
    private String objectName;

    @OneToMany(fetch = FetchType.EAGER, mappedBy = "object")
    private Set<Investment> investments;


    public int getObjectId() {
        return objectId;
    }

    public void setObjectId(int objectId) {
        this.objectId = objectId;
    }

    public String getObjectName() {
        return objectName;
    }

    public void setObjectName(String objectName) {
        this.objectName = objectName;
    }

    public Set<Investment> getInvestments() {
        return investments;
    }

    public void setInvestments(Set<Investment> investments)
{
        this.investments = investments;
    }

    @Override
    public String toString() {
        return "Object{" +
                "object_id='" + objectId + '\'' +
                ", object_name=" + objectName +
                '}';
    }
}
```

Клас Message.java

```
package logic.table;

import javax.persistence.*;
import java.util.Date;

/**
 * @author Dolinniy
 * @version 1.0
 *          <p>
 *          Клас є сутністю, яка використовується для
моделювання об'єкту Message
 *          у відповідній таблиці бази даних
 */
@Entity
@Table(name = "message")
@NamedQuery(name = "Message.getAll", query = "SELECT c from
Message c")
public class Message {
    @Id
    @Column(name = "message_id")
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int messageId;
    @Column(name = "message_subject")
    private String messageSubject;
    @Column(name = "message_time")
    private java.util.Date messageTime;
    @Column(name = "message_text")
    private String messageText;

    @ManyToOne(fetch = FetchType.EAGER, cascade =
{CascadeType.MERGE, CascadeType.PERSIST,
CascadeType.REFRESH}, optional = false)
    @JoinColumn(name = "message_sender_profile_id", nullable
= false)
    private Profile sender;

    @ManyToOne(fetch = FetchType.EAGER, cascade =
```

```
{CascadeType.MERGE, CascadeType.PERSIST,
CascadeType.REFRESH}, optional = false)
    @JoinColumn(name = "message_recipient_profile_id",
nullable = false)
    private Profile recipient;

    public String getMessageSubject() {
        return messageSubject;
    }

    public void setMessageSubject(String messageSubject) {
        this.messageSubject = messageSubject;
    }

    public Date getMessageTime() {
        return messageTime;
    }

    public void setMessageTime(Date messageTime) {
        this.messageTime = messageTime;
    }

    public String getMessageText() {
        return messageText;
    }

    public void setMessageText(String messageText) {
        this.messageText = messageText;
    }

    public Profile getSender() {
        return sender;
    }

    public void setSender(Profile sender) {
        this.sender = sender;
    }

    public Profile getRecipient() {
        return recipient;
    }

    public void setRecipient(Profile recipient) {
        this.recipient = recipient;
    }

    @Override
    public String toString() {
        return "Message{" +
                "message_id='" + messageId + '\'' +
                "message_subject='" + messageSubject + '\''
+
                ", message_time=" + messageTime +
                ", message_text=" + messageText +
                ", message_sender=" + sender.getProfileId()
+
                ", message_recipient=" +
recipient.getProfileId() +
                '}';
    }
}
```

Клас Investment.java

```
package logic.table;

import javax.persistence.*;
import java.util.Date;

/**
 * @author Dolinniy
 * @version 1.0
 *          <p>
 *          Клас є сутністю, яка використовується для
моделювання об'єкту Investment
 *          у відповідній таблиці бази даних
 */
@Entity
@Table(name = "investment")
@NamedQuery(name = "Investment.getAll", query = "SELECT c
from Investment c")
public class Investment {
    @Id
    @Column(name = "investment_id")
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int investmentId;

    @Column(name = "investment_contract_number")
    private int investmentContractNumber;

    @ManyToOne(fetch = FetchType.EAGER, cascade =
{CascadeType.MERGE, CascadeType.PERSIST,
CascadeType.REFRESH}, optional = false)
    @JoinColumn(name = "object_id", nullable = false)
    private Object object;

    @ManyToOne(fetch = FetchType.EAGER, cascade =
{CascadeType.MERGE, CascadeType.PERSIST,
CascadeType.REFRESH}, optional = false)
    @JoinColumn(name = "profile_id", nullable = false)
    private Profile profile;

    public int getInvestmentId() {
        return investmentId;
    }

    public void setInvestmentId(int investmentId) {
        this.investmentId = investmentId;
    }

    public int getInvestmentContractNumber() {
        return investmentContractNumber;
    }

    public void setInvestmentContractNumber(int
investmentContractNumber) {
        this.investmentContractNumber =
investmentContractNumber;
    }

    public Object getObject() {
        return object;
    }

    public void setObject(Object object) {
        this.object = object;
    }

    public Profile getProfile() {
        return profile;
    }

    public void setProfile(Profile profile) {
        this.profile = profile;
    }

    @Override
    public String toString() {
        return "Investment{" +
                "investment_id=" + investmentId + '\'' +
                "investment_contract_number=" +
investmentContractNumber + '\'' +
                ", object_date=" + object.getObjectId() +
                ", profile_id=" + profile.getProfileId() +
                '}';
    }


}
```

Клас Comment.java

```
package logic.table;

import javax.persistence.*;
import java.util.Date;

/**
 * @author Dolinniy
 * @version 1.0
 *          <p>
 *          Клас є сутністю, яка використовується для
моделювання об'єкту Comment
 *          у відповідній таблиці бази даних
 */
@Entity
@Table(name = "comment")
@NamedQuery(name = "Comment.getAll", query = "SELECT c from
Comment c")
public class Comment {
    @Id
    @Column(name = "comment_id")
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int commentId;
    @Column(name = "comment_branch_of_forum")
    private String commentBranchOfForum;
    @Column(name = "comment_time_of_change")
    private java.util.Date commentTimeOfChange;
    @Column(name = "comment_text")
    private String commentText;


    @ManyToOne(fetch = FetchType.EAGER, cascade =
{CascadeType.MERGE, CascadeType.PERSIST,
CascadeType.REFRESH}, optional = false)
    @JoinColumn(name = "profile_id", nullable = false)
    private Profile profile;

    public int getCommentId() {
        return commentId;
    }

    public void setCommentId(int commentId) {
        this.commentId = commentId;
    }

    public String getCommentBranchOfForum() {
        return commentBranchOfForum;
    }

    public void setCommentBranchOfForum(String
commentBranchOfForum) {
        this.commentBranchOfForum = commentBranchOfForum;
    }
```

```
    public Date getCommentTimeOfChange() {
        return commentTimeOfChange;
    }

    public void setCommentTimeOfChange(Date
commentTimeOfChange) {
        this.commentTimeOfChange = commentTimeOfChange;
    }

    public String getCommentText() {
        return commentText;
    }

    public void setCommentText(String commentText) {
        this.commentText = commentText;
    }

    public Profile getProfile() {
        return profile;
    }

    public void setProfile(Profile profile) {
        this.profile = profile;
    }


    @Override
    public String toString() {
        return "Comment{" +
                "comment_id='" + commentId + '\'' +
                ", comment_branch=" + commentBranchOfForum +
                ", comment_time=" + commentTimeOfChange +
                ", comment_text=" + commentText +
                ", profile_id=" + profile.getProfileId() +
                '}';
    }
}
```

Клас Category.java

```
package logic.table;

import javax.persistence.*;
import java.util.Set;

/**
 * @author Dolinniy
 * @version 1.0
 *          <p>
 *          Клас є сутністю, яка використовується для
моделювання об'єкту Category
 *          у відповідній таблиці бази даних
 */
@Entity
@Table(name = "category")
@NamedQuery(name = "Category.getAll", query = "SELECT c from
Category c")
public class Category {
    /**
     *
     */
    @Id
    @Column(name = "category_id")
    @GeneratedValue(strategy = GenerationType.AUTO)

    private int categoryId;

    @Column(name = "category_name")
    private String categoryName;


    @OneToMany(fetch = FetchType.EAGER, mappedBy =
"category")
    private Set<Profile> profiles;

    public Set<Profile> getProfiles() {
        return profiles;
    }

    public void setProfiles(Set<Profile> profiles) {
        this.profiles = profiles;
    }

    public int getCategoryId() {
        return categoryId;
    }

    public void setCategoryId(int categoryId) {
        this.categoryId = categoryId;
    }

    public String getCategoryName() {
        return categoryName;
    }

    public void setCategoryName(String categoryName) {
        this.categoryName = categoryName;
    }

    @Override
    public String toString() {
        return "Category{" +
                "category_id='" + categoryId + '\'' +
                ", category_name=" + categoryName +
                '}';
    }
}
```