Лабораторна робота №3-4

Мета: ознайомлення з основними принципами структурного моделювання програмнихсистемта побудови класів як основних структурних одиниць програмного забезпечення. Закріплення практичних навичок побудови структурних діаграм (зокрема UML-діаграмкласів)та аналізу взаємозв'язків між класами у процесі проектування програмної системи.

Завдання №3. Побудова діаграми класів (Domain Model)

Завдання №3.1 Перелік основних сутностей

Для платформи "TransferAnalytics" виділено наступні сутності:

- User (Користувач): Єдиний клас для користувачів системи. Звичайні користувачі мають роль user, вони можуть переглядати трансфери та вести Watchlist.
- Admin (Адміністратор): Успадковує User. Має додаткові права (модерація, підтвердження трансферів).
- Player (Гравець): Головний об'єкт аналітики.
- Club (Клуб): Організація, що володіє гравцями.
- Transfer (Трансфер): Сутність, що фіксує перехід гравця.
- Watchlist (Список обраного): Дозволяє будь-якому користувачу (User) стежити за гравцем.
- ScoutReport (Скаутський звіт): Детальний аналіз гравця.

Завдання №3.2. Ідентифікація атрибутів класів

Таблиця 3.1. Атрибути класів предметної області

Клас	Атрибут	Тип	Опис
User	id	INT (PK)	Унікальний ID
	username	VARCHAR	Логін
	email	VARCHAR	Email
	password_hash	VARCHAR	Хешований пароль
	role	ENUM	Роль ('user', 'scout', 'admin')
	created_at	TIMESTAMP	Дата реєстрації
Player	id	INT (PK)	ID гравця
	club_id	INT (PK)	Поточний клуб
	name	VARCHAR	ПІБ гравця
	position	ENUM	Позиція (GK, DEF, MID, FWD)
	market_value	DECIMAL	Ринкова вартість
Club	id	INT (PK)	ID клубу
	league_id	INT (FK)	ID ліги
	name	VARCHAR	Назва клубу
	budget	DECIMAL	Бюджет

Transfer	id	INT (PK)	Ідентифікатор трансферу
	player_id	INT (FK)	Гравець
	from_club_id	INT (FK)	Клуб-продавець
	to_club_id	INT (FK)	Клуб-покупець
	fee	DECIMAL	Сума угоди
	status	ENUM	Статус ('rumor', 'done')
Watchlist	id	INT (PK)	ID запису
	user_id	INT (FK)	Користувач
	player_id	INT (FK)	Гравець
ScoutReport	id	INT (PK)	ID звіту
	content	JSON	Дані звіту (фізика, тактика)

Таблиця 3.2. Основні методи класів

Клас	Метод	Повертає	Опис
User	Register(email, password)	void	Реєстрація нового користувача
	Login(email, password)	String	Авторизація (повертає токен)
	EditProfile(data)	void	Оновлення особистих даних
Player	UpdateValue(newValue)	void	Зміна ринкової вартості гравця
	GetTransferHistory()	List	Отримання історії переходів
Club	GetSquad()	List	Отримання списку гравців
	MakeOffer(player,amount)	Transfer	Створення пропозиції про трансфер
Transfer	SetStatus(newStatus)	void	Зміна статусу
	UpdateFee(newFee)	void	Зміна суми трансферу
Watchlist	Add(player_id)	void	Додавання гравця в обране
	Remove(player_id)	void	Видалення зі списку
ScoutReport	Create(player_id, data)	void	Створення нового звіту
	Publish()	void	Публікація звіту

Завдання №3.3. Діаграма класів предметної області

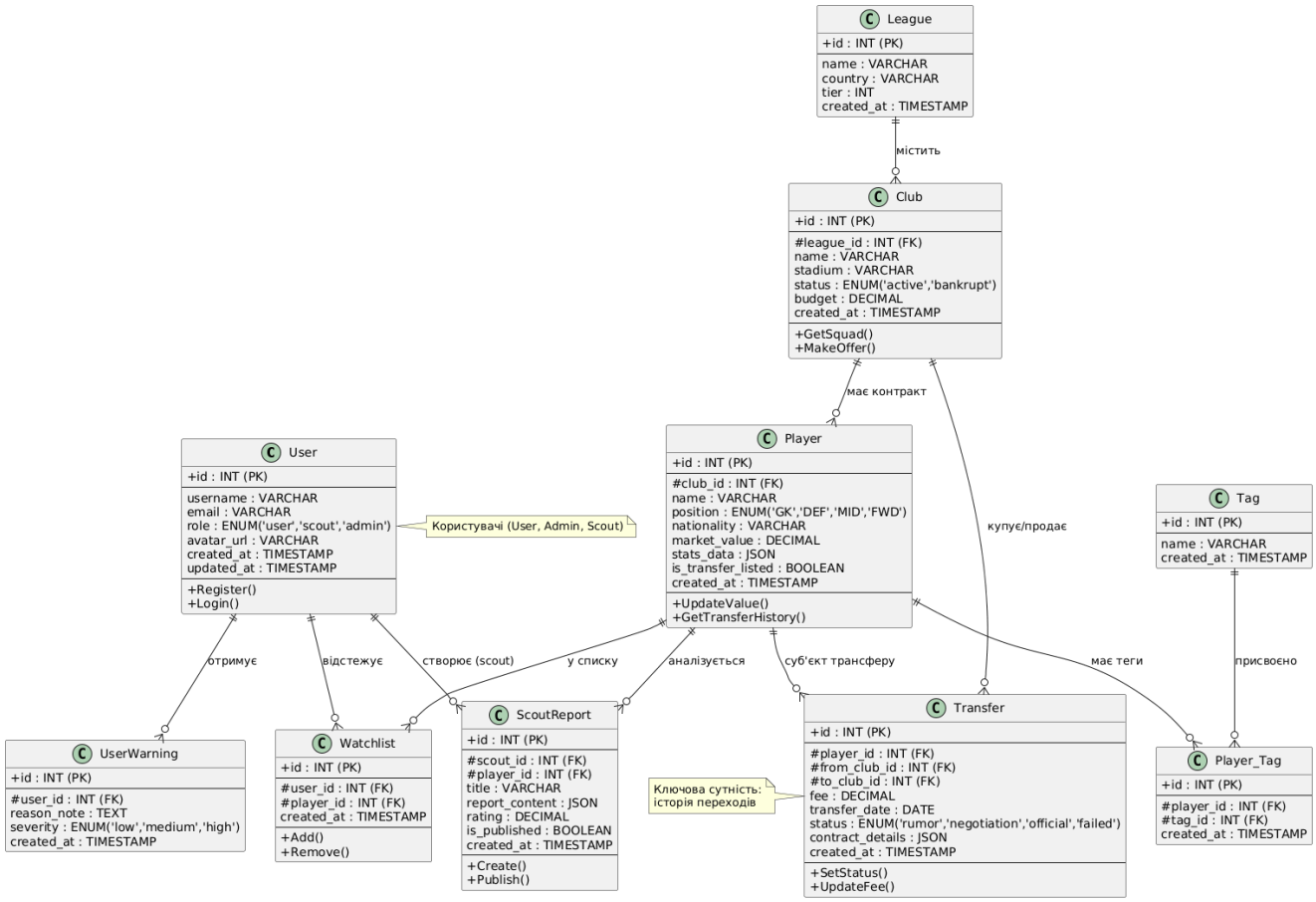

Рис.3.1. Діаграма класів предметної області

Завдання №4. Побудова діаграми класів реалізації (Патерни проектування)

Для реалізації архітектури системи використано наступні патерни проектування :

- **Model-View-Controller (MVC):** Забезпечує розділення логіки.
  - Controller (TransferController) обробляє HTTP-запити.
  - Service (TransferService) містить бізнес-логіку.
  - Repository (TransferRepository) взаємодіє з базою даних.
- **Factory Method (Фабричний метод):**
  - Клас TransferFactory використовується для створення об'єктів трансферу різних типів (оренда, покупка, вільний агент), інкапсулюючи логіку ініціалізації.
- **State (Стан):**
  - Інтерфейс ITransferState та класи NegotiationState, SignedState дозволяють об'єкту Transfer змінювати свою поведінку залежно від поточного етапу угоди (переговори, медогляд, підписання), уникаючи великих конструкцій switch-case.

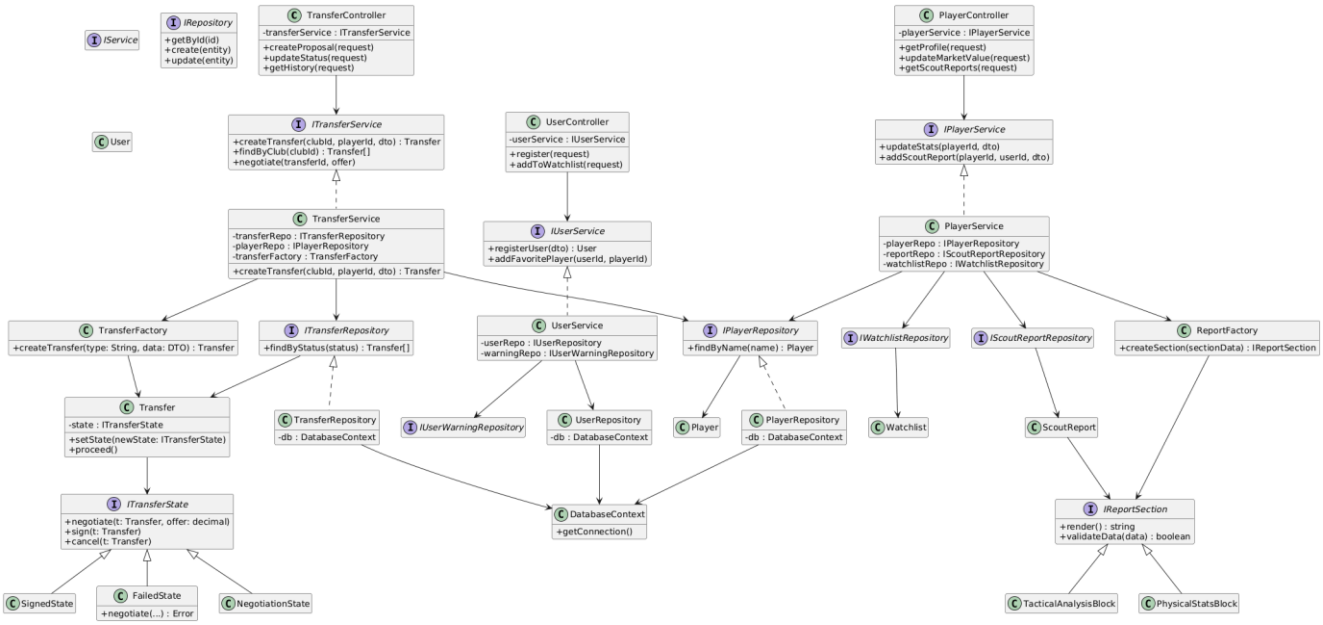

Рис.4.1. Діаграма класів реалізації з патернами проектування

Висновок: в ході виконання лабораторної роботи було побудовано діаграму класів предметної області, визначено атрибути та методи основних сутностей (User, Player, Club, Transfer). Також розроблено діаграму реалізації, де застосовано патерни проектування (MVC, Factory, State) для забезпечення гнучкості та масштабованості системи.

Репозиторій: https://github.com/Svistaniuk/SMAA