МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КПІ»
НАВЧАЛЬНО-НАУКОВИЙ КОМПЛЕКС «ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ»

Кафедра математичних методів системного аналізу

Мурга Микола Олексійович

# ОРГАНІЗАЦІЯ БАЗ ДАНИХ ТА ЗНАНЬ

Методичні вказівки до виконання курсової роботи
для студентів напряму підготовки
6.050101 – «Комп’ютерні науки»

Київ - 2015

УДК 004.6

Рекомендовано Вченою радою Навчально-наукового комплексу «Інститут прикладного системного аналізу» Національного технічного університету України «Київський політехнічний інститут» від.

Автор: ***Мурга Микола Олексійович***, к.т.н., старший викладач кафедри математичних методів системного аналізу ННК «ІПСА» НТУУ «КПІ».

Відповідальний редактор: ***Єфремов Костянтин Вікторович***, директор світового центру даних з геоінформатики та сталого розвитку, старший викладач кафедри автоматизації проектування енергетичних процесів та систем ТЕФ НТУУ «КПІ».

Рецензенти:

***Зайченко Юрій Петрович***, д.т.н., проф., професор кафедри математичних методів системного аналізу ННК «ІПСА» НТУУ «КПІ»

***Теленик Сергій Федорович***, д.т.н., проф., завідувач кафедри автоматики та керування в технічних системах ФІОТ НТУУ «КПІ»

Мурга М.О. Організація баз даних та знань: методичні вказівки до виконання курсової роботи з курсу для студентів напряму підготовки 6.050101 – «Комп’ютерні науки». – К.: кафедра ММСА ННК «ІПСА» НТУУ «КПІ», 2014. – 72 с.

У методичних вказівках визначається структура процесів виконання та захисту курсової роботи по дисципліні та даються детальні роз’яснення з правил оформлення звіту по ній, які супроводжуються наочними прикладами.

**ЗМІСТ**

**ВСТУП…….... 5**

**РОЗДІЛ 1. СТРУКТУРА КУРСОВОЇ РОБОТИ…… 6**

1.1. Структура процесу розробки системи……. 7

1.2. Структура процесу захисту курсової роботи…….. 8

1.3. Структура процесу оцінювання роботи…….. 9

**РОЗДІЛ 2. ПРАВИЛА ОФОРМЛЕННЯ ЗВІТУ З КУРСОВОЇ РОБОТИ.. 11**

2.1.Структура звіту з курсової роботи…… 11

2.2.Правила оформлення елементів звіту з курсової роботи……... 12

2.2.1. Титульний аркуш……... 12

2.2.2. Вступ…….. 12

2.2.3. Основна частина…… 13

2.2.3.1. Аналіз предметної галузі та формулювання системних вимог до додатку баз даних…… 13

2.2.3.2. Структурна схема додатку баз даних……. 21

2.2.3.3. Обґрунтування вибору мови програмування, СКБД та технологій для реалізації додатку баз даних…….. 24

2.2.3.4. Схема розгортання додатку баз даних……... 27

2.2.3.5. Структура баз(и) даних…… 30

2.2.3.6. Діаграма класів додатку баз даних……. 45

2.2.3.7. Обґрунтування доцільності та ефективності запитів до СКБД в додатку баз даних…… 49

2.2.3.8. Опис інтерфейсу з користувачем…… 62

2.2.3.9. Приклади використання додатку баз даних…….. 63

2.2.4. Висновки по роботі…… 64

2.2.5. Критерії оцінювання роботи……. 64

2.2.6. Процес захисту роботи…….. 64

**ВИСНОВКИ……... 66**

**ЛІТЕРАТУРА…… 67**

**ДОДАТОК А. ТИТУЛЬНИЙ ЛИСТ ЗВІТУ З КУРСОВОЇ РОБОТИ 68**
**ДОДАТОК Б. КРИТЕРІЇ ОЦІНЮВАННЯ РОБОТИ……... 69**
**ДОДАТОК В. ПРОЦЕС ЗАХИСТУ РОБОТИ…… 70**
**ДОДАТОК Д. СПИСОК ТЕМ КУРСОВИХ РОБІТ…….. 72**

# ВСТУП

При виконанні курсової роботи з дисципліни «Організація баз даних та знань», яка викладається для бакалаврів напрямку 6.050101 – Комп'ютерні науки ННК «ІПСА» НТУУ «КПІ», у студентів виникає багато складнощів та питань, більша частина з яких є типовими. Окрім того, роз'яснення цих питань можна провести системно, поєднавши їх у логічну послідовність. Саме цей факт визначає необхідність створення даних методичних вказівок.

Об'єктом викладення вказівок є додатки баз даних.

Предметом викладення є зміст процесів виконання та захисту курсової роботи, в якій відбувається розробка додатку баз даних.

Метою викладення є викладення правил та рекомендацій по проходженню процесів виконання та захисту курсової роботи.

Методичні вказівки складаються з двох розділів та чотирьох додатків.

В першому розділі, який має назву «Структура курсової роботи» даються роз'яснення та правила проходження процесів виконання та захисту курсової роботи.

В другому розділі, який називається «Правила оформлення звіту з курсової роботи» дається детальний опис процесів розробки додатку баз даних та оформлення звіту з курсової роботи. Викладення в даному розділі супроводжується наочними прикладами.

Додаток А роботи, який називається «Титульний лист звіту з курсової роботи», містить форму зі змістом Титульного листа звіту, Додаток Б, який називається «Критерії оцінювання роботи» містить правила, за якими відбувається оцінювання курсової роботи студента на ту, чи іншу оцінку. У звіті з курсової роботи студент підписує зміст цього листа, підтверджуючи факт ознайомлення та погодження з ним. Додаток В, який називається «Процес захисту роботи» є спробою формалізації даного процесу та містить основні його етапи, про які студент та викладач вносить відповідні помітки. В Додатку Д, який називається «Список тем курсових робіт», наведено теми, серед яких студент може обрати одну у відповідності до правил вибору, наведених в тому ж додатку.

# РОЗДІЛ 1

## СТРУКТУРА КУРСОВОЇ РОБОТИ

Виконання курсової роботи з дисципліни полягає у проходженні наступних етапів:

1. Вибір теми курсової роботи.
2. Проектування системи для обраної предметної галузі, яка містить у своєму складі певну СКБД, яка керує базою даних цільового додатку.
3. Вибір мов та технологій реалізації спроектованої системи.
4. Реалізація функцій спроектованої системи у вигляді додатку баз даних.
5. Оформлення звіту з курсової роботи, в якому детально описано структуру розробленої системи та наведено приклади її використання.
6. Захист курсової роботи

Тема курсової роботи може бути обрана з Додатку Д. В цьому разі, номер теми, яку повинен опрацювати студент, визначається правилом, викладеним в Додатку. Також студент може обрати собі тему для виконання роботи самостійно, але узгодивши її попередньо з викладачем.

Викладач з дисципліни за бажанням студента може надати консультації з першого та другого з зазначених пунктів. Тобто, викладач може допомогти спроектувати систему та обрати засоби її реалізації.

Реалізація спроектованої системи є обов’язком виключно студента, оскільки складає основу мети курсової роботи. При виникненні питань у процесі реалізації системи, студент повинен користуватися документацією з відповідних проблемних для нього технологій. Викладач може зорієнтувати студента у напрямках пошуку шляхів розв’язання проблеми реалізації, однак розв’язувати проблему замість студента він не зобов’язаний.

Захист курсової роботи є одним із етапів її виконання та полягає у здачі звіту з неї та демонстрацію програмного додатку баз даних до дих пір, доки не будуть усунені всі зауваження викладача, який її приймає.

## 1.1. Структура процесу розробки системи

Процес розробки системи полягає у проходженні наступних етапів:

1. Виділення основних функцій системи та їх формалізація.
2. Структурування виділених функцій в окремих структурних блоках системи.
3. Розмежування структурних блоків системи на їх пакети: користувацької сторони системи, сторони основних обчислень (наприклад, серверної) та бази даних системи.
4. Вибір технологій, які будуть реалізовувати виділені пакети та мов для виділених структурних блоків.
5. Детальне проектування виділених структурних блоків системи на основі результатів попередніх кроків.
6. Реалізація структурних блоків системи.
7. Тестування структурних блоків системи, її пакетів та структури в цілому.
8. Отримання акту про впровадження розробленої системи у реальний комерційний процес є бажаною.

При аналізі предметної галузі, проектуванні системи та її реалізації повинен бути використаний виключно об’єктно-орієнтований підхід.

Для проектування системи доцільно використовувати мову UML, реалізовану в будь-якому її case-засобі (наприклад, Enterprise Architect). Логічну та фізичні схеми баз даних можна проектувати як у тому самому case-засобі, так і спеціалізованих додатках (наприклад, ERWin).

Формальною ознакою, яка є необхідною для позитивної оцінки роботи є наявність в проекті системи мінімум десяти ефективних класів та десяти

ефективних сутностей для подальшого утворення відповідних таблиць. Створення штучних (недоцільних) класів та сутностей є підставою для зниження оцінки за курсову роботу.

Критеріями ефективності проекту системи є:

1. Мінімум десять доцільних класів та сутностей.
2. Наявність ієрархій в структурі класів (можливо і в сутностей).
3. Модульність системи.
4. Використання принципу поліморфізму.
5. Ефективний вибір доменів для атрибутів сутностей.
6. Ефективна нормалізація таблиць бази.
7. Ефективне використання обмежень цілісності в схемі бази.
8. Дружній користувацький інтерфейс.

Система може бути реалізована як у вигляді додатку для робочого стола, web-додатку чи мобільного додатку, так і у будь-якій їх комбінації.

Слід відмітити, що більшість з сучасних технологій програмування мають написані класи, які студент може використати при реалізації системи. Такі класи не входять в перелік з десяти класів, які вимагаються від студента для реалізації в роботі. Класи, які наслідуються від стандартних класів технологій, теж не входять в даний перелік з десяти класів. Сутністні класи не враховуються при підрахунку кількості класів, реалізованих студентом.

Для допуску до захисту курсової роботи, програмний додаток для робочого столу повинен бути повністю робочим, запускатися без IDE, за допомогою якої він був написаний, в «один-два кліки» (.exe-файли, .jar-архіви з прописаним класом, який містить main в маніфесті, .bat і .sh-скрипти). Web-додаток дозволяється демонструвати на одній робочій станції. Мобільний додаток повинен запускатися в «один клік».

## 1.2. Структура процесу захисту курсової роботи

Алгоритм захисту курсової роботи з дисципліни наступний:

1. Оформлений у відповідності до вимог та правил, звіт з курсової роботи передається на перевірку викладачу.

2. Викладач перевіряє звіт та вказує на аркушах «Процес захисту роботи», який наведений у Додатку В, у розділі «Зауваження по звіту з курсової роботи», свої зауваження.

3. У випадку, коли зауваження не заперечують можливість функціонування програмного додатку, студент демонструє викладачу роботу додатку. Свої зауваження по ньому він заносить у розділ «Зауваження по програмному додатку» в аркуші «Процес захисту роботи».

4. Якщо зауваження у викладача відсутні, або сплив термін захисту курсової роботи, робота студента оцінюється. В іншому випадку – на етап 1 алгоритму.

Друкується два екземпляри аркушів «Процес захисту роботи», один з яких надається викладачу, а інший зберігається разом з роботою у вигляді її передостанньої та останньої сторінки.

При виконанні умов завершення процесу захисту курсової роботи, відбувається її оцінювання у відповідності до розділу «Порядок оцінювання курсової роботи» даних методичних рекомендацій.

Перед оцінюванням роботи, студент повинен принести, збережений на CD-R диску, програмний додаток у вигляді, завдяки якому його було б легко розгорнути та використати. На даному диску студент власноруч вносить поточну дату, пише свої прізвище, ім’я, по батькові, групу та ставить підпис.

Після оцінювання дипломної роботи, викладачем робляться відповідні до Додатку В записи в аркушах «Процес захисту роботи» та робота вважається захищеною.

### 1.3. Структура процесу оцінювання роботи

При першій здачі звіту по курсовій роботі на перевірку, студент на третьому з кінця аркуші, із змістом, наведеним в Додатку Б, робить рукою

запис «З правилами оцінювання курсової роботи ознайомлений» та ставить поточну дату і свій підпис.

Як вже було зазначено вище, процес оцінювання роботи починається в момент, коли у викладача, який приймає курсову роботу, відсутні зауваження по змісту звіту з курсової роботи та по програмному додатку, або у разі, якщо час, відведений під захист курсової роботи сплив.

Оцінювання відбувається викладачем у відповідності до «Критеріїв оцінювання роботи».

Таким чином, із вказаного вище випливає, що робота, оцінена до моменту останньої консультації, або двох академічних годин, відведених для захисту, завжди оцінюється на «А – відмінно», а після – на визначену викладачем оцінку. Отже, очевидно, що, слідуючи даній процедурі, об’єктивних причин у студента оскаржувати результати оцінювання його курсової роботи викладачем немає.

Діючи у відповідності до розділу 10.1 пункт 2 «Положення про організацію навчального процесу в НТУУ «КПІ»», викладач може прийняти рішення дозволити студенту усунути його зауваження після завершення часу, відведеного під виконання та захист курсової роботи. Але це рішення приймається викладачем індивідуально для кожного студента та є виключним правом викладача, яке дозволяє йому вирішити, поставлені перед ним дидактичні завдання. У загальному випадку студент повинен пройти процедури захисту та оцінювання роботи до моменту завершення часу відведених під них.

У випадку, якщо студент не розпочав процес захисту до моменту часу, відведеного для виконання та оцінювання роботи, в відомості з курсової роботи, яка подається в деканат, робиться помітка «Не з’явився» та виставляється початкова кількість балів – 0.

## РОЗДІЛ 2

# ПРАВИЛА ОФОРМЛЕННЯ ЗВІТУ З КУРСОВОЇ РОБОТИ

### 2.1. Структура звіту з курсової роботи

Звіт з курсової роботи по дисципліні складається з наступних частин:

1. Титульний аркуш.
2. Вступ.
3. Основна частина, яка складається з:

3.1. Аналіз предметної галузі та формулювання системних вимог до додатку баз даних.

3.2. Структурна схема додатку баз даних.

3.3. Обґрунтування вибору мови програмування, СКБД та технологій для реалізації додатку баз даних.

3.4. Схема розгортання додатку баз даних.

3.5. Структура баз(и) даних.

3.6. Діаграма класів додатку баз даних.

3.7. Обґрунтування доцільності та ефективності запитів до СКБД в додатку баз даних.

3.8. Опис інтерфейсу з користувачем.

3.9. Приклади використання додатку баз даних.

4. Висновки по роботі.
5. Критерії оцінювання роботи.
6. Процес захисту роботи.

Зміст звіту з курсової роботи виконується в науковому стилі. Він повинен точно висвітлювати предмет кожного елемента звіту, бути чітким та лаконічним. Наявність у звіті інформації, яка не стосується його предмету, може послужити підставою для зниження оцінки за курсову роботу.

Звіт з курсової роботи повинен бути виконаний за використання шрифту Times New Roman. Розмір шрифту – 14 пт. Відступ між рядками – 1,5 пт. Типове вирівнювання – по ширині листа. Відступи на сторінці – 2 см. зверху та знизу, 3 см. зліва та 1 см. справа.

Кожна сторінка звіту з курсової роботи нумерується. Номер сторінки проставляється у її правому нижньому куті.

## 2.2. Правила створення елементів звіту з курсової роботи

### 2.2.1. Титульний аркуш

Титульний аркуш звіту є його першою сторінкою. На титульному аркуші номер сторінки не проставляється.

Шаблон оформлення титульного аркушу звіту з курсової роботи можна знайти в Додатку А.

### 2.2.2. Вступ

У вступі необхідно навести:

1. Короткий опис (два-три речення) особливостей проблемної галузі теми курсової роботи.
2. Об’єкт, предмет та мету роботи.
3. Описати властивості програмного додатку баз даних, який буде розроблений.
4. Навести та описати структуру звіту з курсової роботи.

Вступ до курсової роботи за своїм обсягом не повинен перевищувати однієї сторінки. У вступі до курсової роботи повинна міститися тільки така інформація, яка дозволяє поверхово зрозуміти її зміст. Використання інформації, яка не сприяє досягненню цієї мети (наприклад, загальні міркування про предметну галузі та її важливість) можуть послужити підставою для зниження оцінки за курсову роботу.

Вступ до курсової роботи виконується після вже створених інших елементів звіту з курсової роботи, оскільки одна з його частин повинна точно охопити їх зміст.

### 2.2.3. Основна частина роботи

Метою створення основної частини звіту з курсової роботи є всебічне висвітлення результатів праці студента над її темою. Таким чином, у звіті студент повинен найбільш точно описати свої здобутки по роботі. Отже, опис моделей, методів та технологій, які не були створені студентом є зайвою інформацією для звіту і викладач може дати студенту зауваження про необхідність її усунення.

#### 2.2.3.1. Аналіз предметної галузі та формулювання системних вимог до додатку баз даних

При аналізі предметної галузі теми курсової роботи, студент повинен зосередити свою увагу на наступних артефактах, які підлягають створенню:

1. Процеси предметної галузі, в яких буде використано додаток баз даних.
2. Функції додатку баз даних в цих процесах.
3. Критерії ефективності виконання виділених функцій додатком баз даних
4. Структура процесів використання додатку баз даних.

Слід зазначити, що правила оформлення звіту з курсової роботи не обмежують студента у виборі нотації для опису своїх результатів. Єдина вимога – щоб ця нотація дозволяла описувати артефакти аналізу, проектування та програмування, які створюються з використанням об’єктно-орієнтованого підходу.

Для прикладів в даних методичних рекомендаціях, буде використано нотацію мови UML 2.0 [1; 2].

В межах даної нотації для створення змісту розділу «Аналіз предметної галузі та формулювання системних вимог до додатку баз даних» доцільно використати діаграми варіантів використання (Use Case diagrams) та діаграми послідовності взаємодій (Sequence diagrams), що відповідають виділеним варіантам використання.

***Виділення основних процесів, в яких буде використано додаток баз даних***

Будь-який програмний додаток створюється з метою спрощення вирішення користувачем його задач в межах деякого процесу, чи групи процесів.

Кожний процес має певну структуру, елементами якої можуть виступати більш детальні процеси. Таким чином, система процесів будь-якої природи має ієрархічну структуру.

Для опису у звіті процесів, в яких буде використано додаток баз даних, достатньо їх перерахувати, давши коротку характеристику та підпорядкованість.

Надалі у методичних рекомендаціях будуть розглядатися приклади для теми «Система підтримки обслуговування складу товарів».

**Приклад 2.2.3.1.1.**

У складі товарів можна виділити такі основні процеси:

1. Прийом нового товару. – Цей процес охоплює питання: оформлення документації на новий товар, перевірка його кількості та (або) якості, реєстрації та розміщення для зберігання.
2. Видача товару. – Цей процес охоплює питання: встановлення його місцезнаходження, оформлення документації та видачу товару.
3. Списання товару. – Цей процес охоплює питання визначення факту невідповідності якості товару його документації, узгодження необхідності списання товару та безпосередньо списання.

*Зауваження*. Очевидно, що у складі товарів існують і інші процеси, наприклад, процес його опалення. Однак програмний додаток баз даних не можливо створити для всіх існуючих процесів. При його створенні необхідно формалізувати лиш процеси, які обумовив замовник додатку. Однак систему завжди необхідно проектувати так, щоб її можна було без зайвих зусиль поширити на будь-які інші процеси. *Кінець зауваження.*

Далі, в виділених процесах, необхідно виділити існуючи дочірні процеси.

Для прикладу обрано процес «Прийом нового товару».

Даний процес включає в свою структуру наступні дочірні процеси:

1. Знаходження відповідної документації, необхідної для оформлення нового товару.
2. Перевірка відповідності товару нормативним вимогам складу та його власній документації.
3. Прийняття та реєстрація товару. – Даний процес включає питання присвоєння товару інвентарного номеру, знаходження йому відповідного місця на складі, розміщення товару, реєстрація власних та юридичних характеристик товару, його номеру та місця товару на складі.

*Кінець прикладу*

В звіті до курсової роботи необхідно описати всі процеси предметної області, що відповідають його темі. Однак зменшення рівня абстрактності процесу пов’язано з ризиком надлишкової деталізації проекту. Для уникнення цієї ситуації, при побудові ієрархії процесів, необхідно користуватися принципом доцільності конкретизації для ознайомлення з структурою предметної галузі.

***Виділення функцій додатку баз даних в цих процесах***

Для опису виділених функцій додатку баз даних зручно використовувати діаграми варіантів використання.

Будь який процес має об’єкт, на зміну стану якого він напрямлений та діючі особи (актори), які змінюють стан об’єкту.

Власне перед початком виділення функцій додатку баз даних, для кожного виділеного процесу необхідно визначити акторів та його об’єкт.

Після цього, для кожного актора необхідно визначити його функції при зміні стану об’єкту.

З набору виділених функцій необхідно виділити частину, яка буде реалізовуватися за допомогою програмного додатку.

Таким чином, правильно говорити не про функції програмного додатку, а про функції, які він допомагає реалізовувати. Але, оскільки питання користувацького інтерфейсу не належать до предмету дисципліни, функції, які програмний додаток допомагає реалізовувати, будуть у подальшому ототожнюватися з його функціями.

**Приклад 2.2.3.1.2.**

Розглядається процес прийняття та реєстрації товару.

На мові UML, та сторона об’єкта з якою безпосередньо взаємодіють актори, називається суб’єктом.

В даному процесі суб’єктом може виступати однойменна з процесом система.

В якості акторів можуть виступати: менеджер та постачальник.

На підставі виділених елементів можна виділити функції акторів в процесі, які зображені на Рис. 2.2.3.1.1..

Як видно з даного рисунку, функція реєстрації товару включає в себе функції: підготовки нормативної документації для реєстрації товару та знаходження товару місця на складі. Всі ці функції знаходяться в компетенції менеджера складу.

*Зауваження*. Отже, тут, як і в структурі процесів, спостерігається ієрархічність. Взагалі кажучи, наявність ієрархій є однією з ознак складності системи. Формалізація таких ієрархій дозволяє спростити роботи з ними. *Кінець зауваження.*

Функція перевірки товару його документації є спільною для обох акторів. І вона полягає у звірці документації товару з фактичним результатом доставки товару Постачальником до складу. Постачальник та Менеджер виконують цю звірку разом та оформлюють документи видачі-прийняття товару.

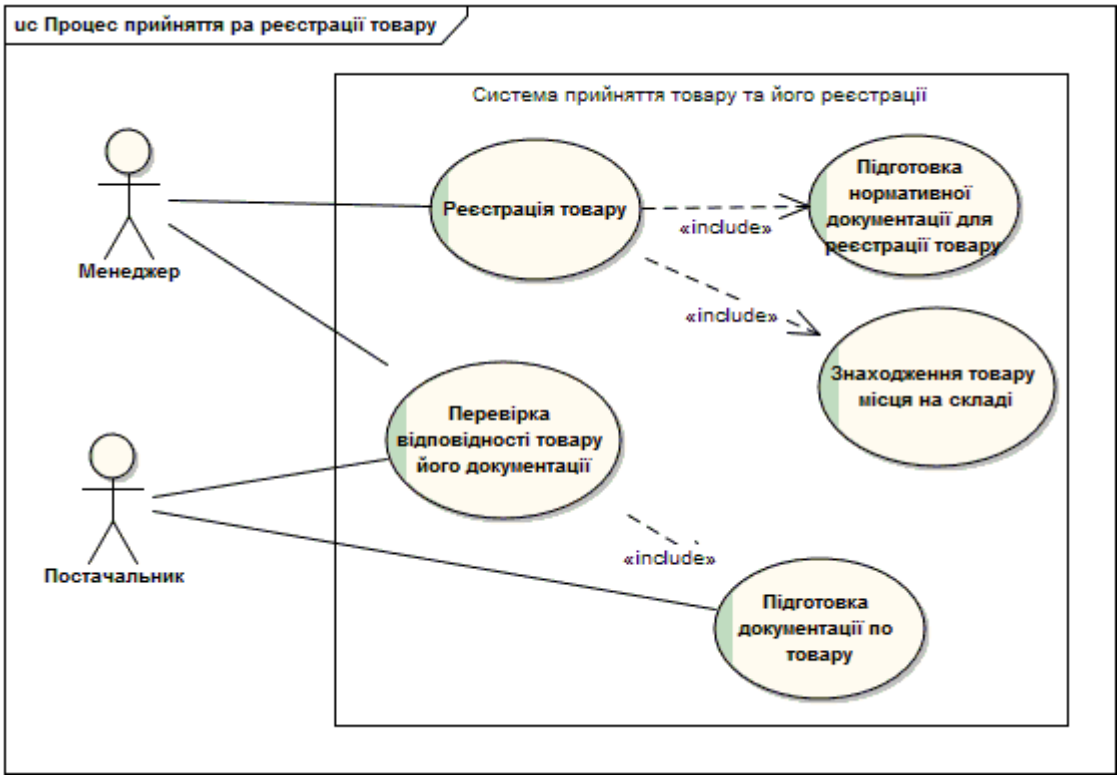


**Рис. 2.2.3.1.1. Діаграма варіантів використання для процесу прийняття товару та його реєстрації**

Як позначено на рисунку, функція перевірки відповідності товару його документації потребує реалізації функції підготовки документації по товару, яка є в межах компетенції Постачальника.

*Кінець прикладу*

Наведену в прикладі формалізацію необхідно виконати для всіх виділених раніше процесів.

Слід відмітити, що процес виділення функцій завжди залежить від проектувальника і, таким чином, читач міг би виділити свої елементи діаграми використання для опису функцій в рамках процесу прикладу.

Вдалий вибір елементів визначає те, наскільки процес буде якісно формалізований та описаний і, як наслідок, дозволить уникнути в майбутньому

необхідності розробки удосконалень програмного додатку, що зменшить його вартість.

***Визначення критеріїв ефективності виконання виділених функцій додатком баз даних***

Очевидно, що ефективність реалізації певної функції додатка баз даних залежить від якості його розробки. Одні додатки можуть реалізовувати функцію краще за інших. Таким чином існує певне відношення переваги на множині додатків і, можливо, навіть не одне. Неоднозначність оцінки ефективності додатку баз даних є наслідком неоднозначності сприйняття результатів його роботи.

Оцінка ефективності реалізації функцій додатками відбувається на основі деяких критеріїв. Аналізуючи значення цих критерії можна прийти до висновку про ефективну роботу додатку, або про необхідність пошуку шляхів підвищення якості його роботи.

Слід зразу відзначити, що при проектуванні реальних додатків баз даних не дуже часто приділяють увагу питанню виділення критеріїв якості реалізації ними функцій. Однак позитивні результати цього процесу все одно слід мати на увазі та все ж його провадити.

**Приклад 2.2.3.1.3.**

В якості прикладу виділення критерія ефективності можна навести приклад для функції «Реєстрація товару».

Тут в якості прикладу можна виділити критерій – кількість помилок при реєстрації на одиницю часу.

При великих значення цього показника можна прийти до висновку про необхідність покращення інтерфейсу користувача додатку баз даних.

*Кінець прикладу*

**Приклад 2.2.3.1.4.**

Також для даної функції критерієм може слугувати критерій – середнє значення відношення загальної кількості часу, затраченого для проведення транзакції реєстрації до кількості часу роботи СУБД.

Великі значення цього критерію свідчать про необхідність оптимізації структури бази

*Кінець прикладу*

В курсовій роботі необхідно виділити критерії ефективності реалізації тільки основних функцій додатку баз даних.

***Структура процесів використання додатку баз даних***

Наступним кроком при проектуванні додатків баз даних є деталізація виділених функцій. Для досягнення цієї мети зручним інструментом є діаграми послідовностей взаємодій (sequence diagrams) мови UML (також може бути використана діаграма комунікацій).

Діаграма послідовностей взаємодій включає в себе об’єкти класів, описує їх час життя та дозволяє зобразити обмін повідомленнями між ними.

Всі діаграми UML можуть бути використані для опису різних рівнів абстракції системи. Але опис всіх рівнів не завжди є доцільним, а лише перевантажує документацію для додатку баз даних.

Об’єкти в діаграмах послідовностей є як реальними об’єктами класів додатку, так і об’єктами реального світу, які використані для покращення розуміння структури виділеної функції.

**Приклад 2.2.3.1.5.**

Нехай для опису обрано функцію «Знаходження товару місця на складі».

На даний момент рівень деталізації опису системи та додатку баз даних, який повинен спростити взаємодію з нею, не є достатнім для опису взаємодій об’єктів класів додатку.

Окрім того, очевидно, що для реалізації даної функції необхідно вимагати певних дій і від менеджера.

Таким чином, для опису функції було виділено об’єкти: менеджер, користувацький інтерфейс, контролер та СКБД. Отже, в даному випадку жоден з об’єктів не є програмним об’єктом додатку баз даних (все ж контролер в деяких технологіях програмування виділяється у вигляді об’єкту. Наприклад, в технології Spring MVC).

Результати взаємодії об’єктів представлені на Рис.2.2.3.1.2.

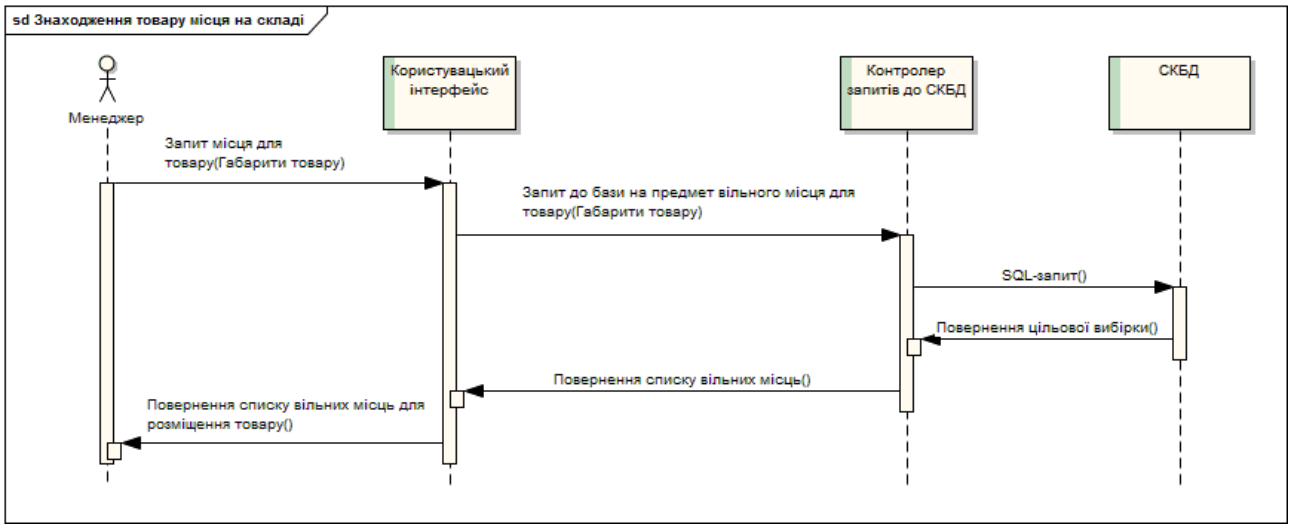


**Рис. 2.2.3.1.2. Діаграма послідовностей для опису функції знаходження товару місця на складі**

В даній діаграмі зображено наступний алгоритм дій для виконання виділеної функції:

1. Менеджер робить запит, передаючи габарити товару користувацькому інтерфейсу.

2. Програмна частина користувацького інтерфейсу перетворює запит користувача у форму придатну для опрацювання контролером запитів до СКБД.

3. Контролер формує оптимізований запит до СКБД на предмет видачі списку вільних місць для товару, що відповідають його габаритам.

4. Отриманий список передається програмній частині користувацького інтерфейсу, яка робить отриманий список доступним для сприйняття та опрацювання менеджером.

*Кінець прикладу*

В даному прикладі функція описана достатньо детально для розуміння її структури. Але подальша деталізація могла б привести до створення зайвих класів.

Процес, приведений у прикладі, необхідно повторити для всіх функцій, які були виділені протягом виконання курсової роботи.

### 2.2.3.2.Структурна схема додатку баз даних

За виділенням та деталізованим описом функцій додатку баз даних повинен слідувати опис його структури. Добре описана структура дозволяє поверхово побачити, що є в реалізованому додатку, а чого в ньому немає.

Для опису структури можна використовувати класичну схемотехнічну нотацію, а можна і мову UML. В даній мові для цього дуже добре підходять або діаграма компонентів, або діаграма композитної структури.

В прикладі будуть використані позначення діаграми компонентів.

**Приклад 2.2.3.2.1.**

Для тієї ж тематики (склад товарів) необхідно побудувати структурну схему.

Для представлення структурної схеми обрано діаграму компонентів мови UML, яка зображена на Рис. 2.2.3.2.1.

Як видно, з даного рисунку, з попередньо складеної схеми функцій системи, в якій міститься додаток баз даних, виділено чотири його структурні блоки: блок графічного інтерфейсу користувача, блок системи реєстрації товару, блок служби транспортування та зберігання товару на складі та блок бази даних складу. З’єднання між блоками представлені у вигляді зв’язків типу «штепсель-розетка».

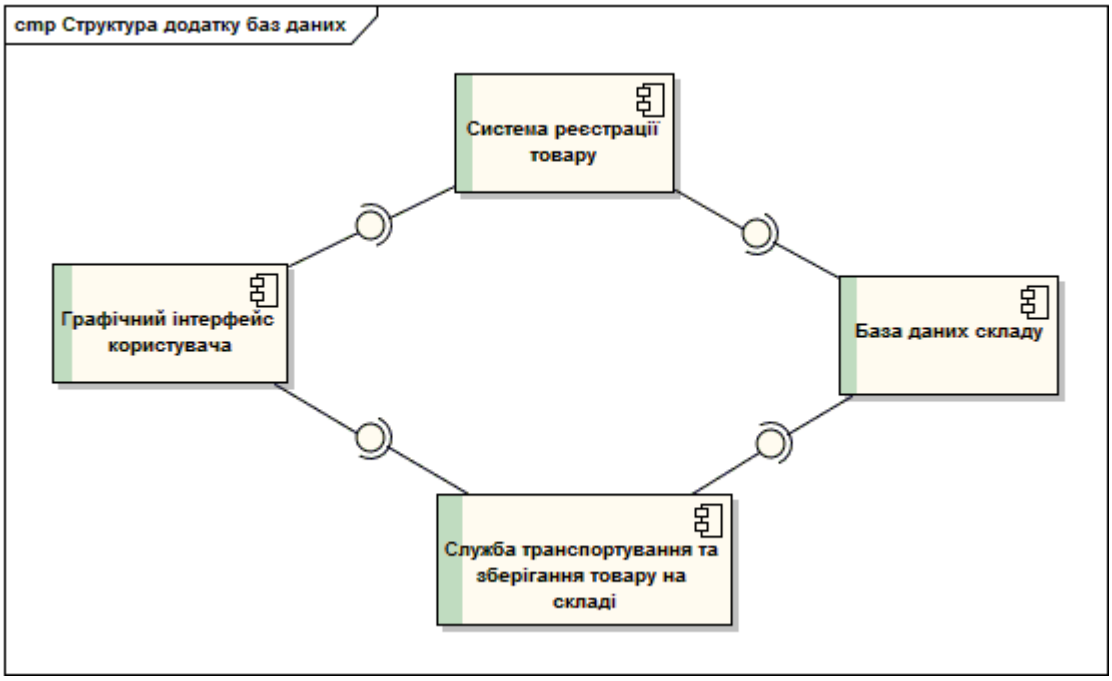


**Рис. 2.2.3.2.1. Структура додатку баз даних**

Блок (компонент – в термінології UML) графічного інтерфейсу користувача використовує сервіси, які пропонують блок системи реєстрації товару та блок служби транспортування та зберігання товару на складі. Тобто, блок графічного інтерфейсу використовує, по суті, методи класів даних блоків як для отримання спеціалізованої інформації з бази, так і для збереження в ній інформації про товар.

Блоки системи реєстрації товару та служби транспортування та зберігання товару на складі використовують базу даних (через СКБД, але про неї на даному етапі не слід писати, оскільки вона буде визначена при виборі технологій реалізації та на діаграмі розгортання)

*Кінець прикладу*

Слід відмітити такий факт: в прикладі була використана фраза: «…методи класів даних блоків…». Таким чином, діаграми компонентів можна вважати попереднім кроком для початку побудови діаграми класів, але її можна починати будувати лише після визначення особливостей реалізації додатку баз даних.

В прикладі зображено два компоненти за виключенням компонент графічного інтерфейсу та бази даних (тобто, в сумі - чотири). Дану кількість

можна вважати прийнятною для курсової роботи. Для роботи достатньо два-три службових компоненти за виключенням графічного інтерфейсу та бази даних.

З діаграми компонентів стає очевидною структура додатку баз даних, але напрямки потоків операцій, що відбуваються всередині процесів залишаються незрозумілими.

Тому опис даного елементу звіту з курсової роботи при необхідності можна доповнити принциповою схемою додатку баз даних (в термінах схемотехніки). В UML для представлення принципової схеми додатку можна використати як діаграми комунікацій.

Для наведеного вище прикладу малювати принципову схему є зайвим, оскільки структурна схема дуже добре описана словесно і в цьому описанні присутні напрямки потоків операцій. Однак для ознайомлення нижче приводиться приклад принципової схеми.

Слід відмітити, що створення зайвих діаграм, які не покращують якість сприйняття змісту звіту, може стати причиною зниження оцінки за курсову роботу.

**Приклад 2.2.3.2.2.**

На Рис. 2.2.3.2.2 Наведена схема додатку баз даних, як альтернатива, розглянутій у попередньому прикладі.

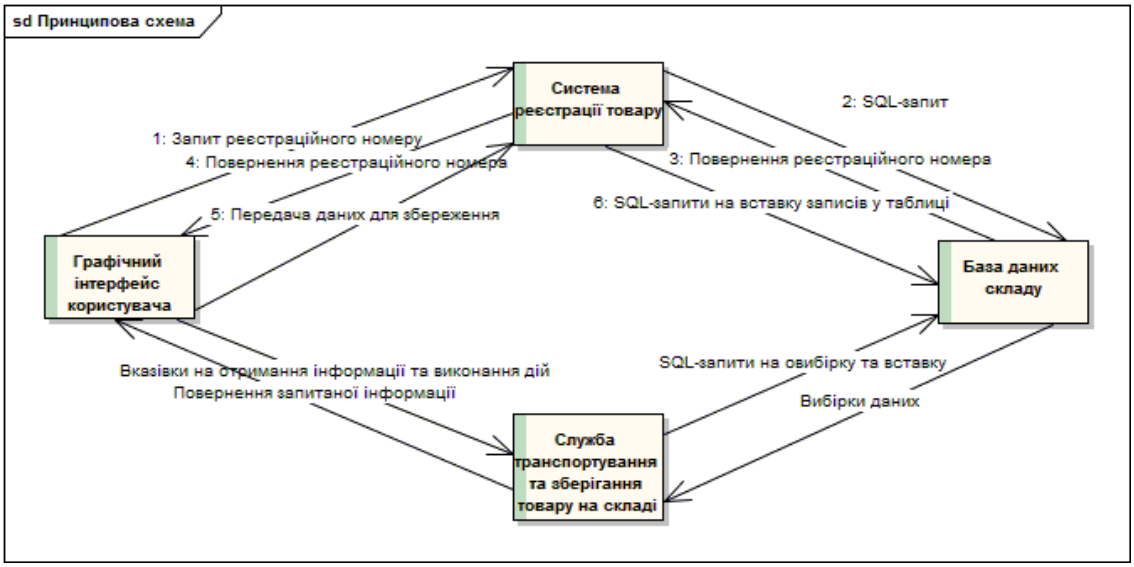


**Рис. 2.2.3.2.2. Принципова схема додатку баз даних**

На справді, коментарі для даної схеми є зайвими, оскільки вся необхідна інформація описана на ній. Можна продублювати її словами, але тоді стає очевидною перевага діаграми компонентів у поверховому описі додатку баз даних.

*Кінець прикладу*

Якщо порівняти два останні малюнки, то можна побачити, що діаграма компонентів є менш перевантаженою. Звісно, в діаграмі взаємодій можна використати двонаправлені взаємозв'язки між блоками, однак тоді повністю втрачається інформативність даної діаграми.

В діаграмі компонентів очевидно, який блок пропонує сервіси, однак не очевидні напрямки потоку дій. Проте супровід описом даної діаграми, не займає надто багато місця і дозволяє подолати цей недолік.

Тому, на думку автора, діаграма компонентів більш підходить для опису загальної структури додатку баз даних, ніж діаграма взаємодій. Але, напевне, це – справа власних уподобань проектувальника.

Головна мета викладення даного елементу звіту з курсової роботи: як можна менший та найбільш інформативний опис загальної структури додатку баз даних.

### 2.2.3.3. Обґрунтування вибору мови програмування, СКБД та технологій для реалізації додатку баз даних

Здебільшого вибір мов та технологій проходить автоматично, спираючись на наявні знання у проектувальника та розробників. Використання нових мов та технологій при розробці системи не завжди доцільно внаслідок необхідних затрат часу та коштів (інколи немалих) на навчання та отримання відповідних сертифікатів та ліцензій.

Однак, в ідеальному випадку, розробка кожної системи потребує для окремих її частин освоєння нових мов та технологій, для відповідності цих

систем сучасним вимогам користувачів та високої обчислювальної ефективності системи.

У ідеального випадка є аргумент на користь використання відомих мов та технологій. Популярним є вислів: «Гарний програміст може програмувати на чому завгодно і з невеликим зусиллям на тому, на чому це б здавалося неможливим». Будь-що можна реалізувати на будь-якій мові програмування, але існування вже готових технологій на окремих мовах зменшує час на розробку високоефективних рішень.

Окрім того, на підставі свого досвіду автор вказівок може виділити наступні базові складові процесу програмування:

1. Неперервний процес перепроектування.
2. Використання ефективних готових рішень (технологій).
3. Розстановка конструкцій мови програмування та технологій у відповідності до послідовностей команд на єдиній алгоритмічній мові.

Слід підкреслити, що автор виділив лише базові (необхідні) складові і реальний процес програмування включає значно більше пунктів. Але навіть другий пункт – використання ефективних готових рішень, веде за собою необхідність вивчення цих рішень і, тому, час на повне оволодіння всіма конструкціями програмування мови в багато разів перевищує час на освоєння її базових конструкцій. Це є причиною того, що більшість компаній або винаймають спеціалізованих програмістів, або самі спеціалізуються на написанні програм на окремих мовах програмування.

Отже, напевно, по даному елементу звіту з курсової роботи сформулювати чіткі правила неможливо, можливо дати лише рекомендації, що і було тут зроблено. Однак, щоб зорієнтуватися, що ж таки вимагається у звіти по даному елементу, наводиться наступний приклад. Але слід пам’ятати, що це – один з можливих прикладів.

**Приклад 2.2.3.3.1.**

Для вибору мов та технологій бажано б було мати діаграму розгортання, однак це є наступним кроком процесу розробки і така послідовність, напевно, є

доцільною оскільки дозволяє розташувати фізичні об'єкти найбільш ефективним образом стосовно задіяння технологій, що планується використати.

Для вибору мов та технологій, насправді, достатньо прийняття рішення по наступних елементах:

1. Тип додатку: настільний, мережевий чи мобільний.
2. Потенційна завантаженість спеціалізованих серверів.
3. Сукупність знань по мовам програмування, технологіям та СКБД, якими володіє колектив розробників на момент початку розробки.
4. Доступні ліцензії.

Для тієї ж теми складу можна зробити припущення, що це великий склад, в якому працюють більше однієї тисячі менеджерів. З цього припущення слідує, що:

1. Внаслідок специфіки своєї роботи, замовники бажали б використовувати як свої мобільні пристрої, так і стаціонарні комп’ютери та ноутбуки. Якщо зробити припущення, що в межах складу організовано Intra-мережу, то цікавим є реалізація веб-додатку, який би зменшив ризик виносу інформації за межі складу та нівелював необхідність встановлення спеціалізованого додатку на кожний пристрій.
2. Далі, необхідно було б централізувати роботу з базою даних для зменшення кількості системних помилок (децентралізація в такій системі є зайвою), тобто можна використати один сервер баз даних. СКБД, що використовується все ж повинна мати можливість розподіляти збереження самих даних на різних хостах. Однак, купляти ліцензії Oracle власник складу не може собі дозволити (припустимо). З цієї причини привабливим рішенням є СКБД PostgreSQL (але, звісно, це рішення не єдине), яка є, до того ж, безкоштовною для комерційного використання.
3. Окрім того, нехай колектив розробників досконало володіють мовою програмування Java. Отже, одним з ефективних виборів в даній ситуації буде вибір технології Spring MVC, а для браузерної частини – Bootstrap

Framework, який вимагає мінімальних знань CSS та JavaScript. Більш того, всі ці рішення є безкоштовними для комерційного використання.

Слід зазначити, що при проектуванні, питанням ліцензій необхідно приділяти велику увагу, оскільки після розробки змінити елементи додатку баз даних може виявитися практично не можливо, а використання неліцензованих елементів може мати важкі юридичні та фінансові наслідки для компанії розробника.

*Кінець прикладу*

### 2.2.3.4.Схема розгортання додатку баз даних

Для продовження процесу проектування, необхідно мати схему фізичного розташування додатку баз даних. Для цієї мети здається більше за інші підходить діаграма розгортання (deployment diagram) UML.

В результаті проходження попередніх кроків проектування, вже відомі кількісний та якісний склад програмних артефактів, які будуть складати додаток баз даних. Конкретизація їх фізичного розміщення в системі дозволить уникнути створення зайвих артефактів діаграм класів та сутностей.

Нижче приводиться приклад опису фізичного розташування елементів додатку баз даних.

**Приклад 2.2.3.4.1.**

*Початок прикладу*

В результаті проходження попередніх кроків проектування, становиться очевидним, що система хостів, на яких будуть функціонувати елементи додатку баз даних складається з наступних одиниць:

- Хост з сервером додатків;
- Хост з сервером баз даних PostgreSQL;
- Хости, які будуть фізично зберігати дані таблиць бази.

Насправді, при достатній потужності серверного хоста, перші два серверні процеси можна розгорнути на одному фізичному хості. Окремі таблиці, які

потенційно не повинні мати дуже велику кількість записів та таблиці, до яких необхідний максимально швидкий доступ, теж можуть бути розміщені на тому самому фізичному хості.

Нехай зроблено припущення, що для розгортання виділено два хости: потужний в обчислювальному плані комп'ютер, що має невеликий обсяг постійної пам'яті (недостатній для розміщення на ньому всіх таблиць бази) та мережеве обладнання, яке спеціалізується на збережені даних і має спеціальні засоби захисту інформації, яка в ньому міститься.

На підставі цього припущення, можливо спроектувати фізичну структуру системи, яка зображена на Рис. 2.2.3.4.1.

*Зауваження.* На рисунку видно, що деталізація до рівня змісту таблиць бази та класів не відбувається, але на підставі цього рисунку можна зробити висновки про фізичну структуру системи. *Кінець зауваження.*

З браузера користувача викликається сторінка index.jsp, яка для представлення front-end додатку використовує Bootstrap Framework (його файли розміщені в постійній пам'яті хоста) та далі відбувається задіяння технології Java Spring MVC.

Для роботи з базою використовується об'єктно-реляційне відображення (основною якого є Hibernate, що задіюється JPA).

Таблиці бази зберігаються на хостах у відповідності до умов, описаних вище.

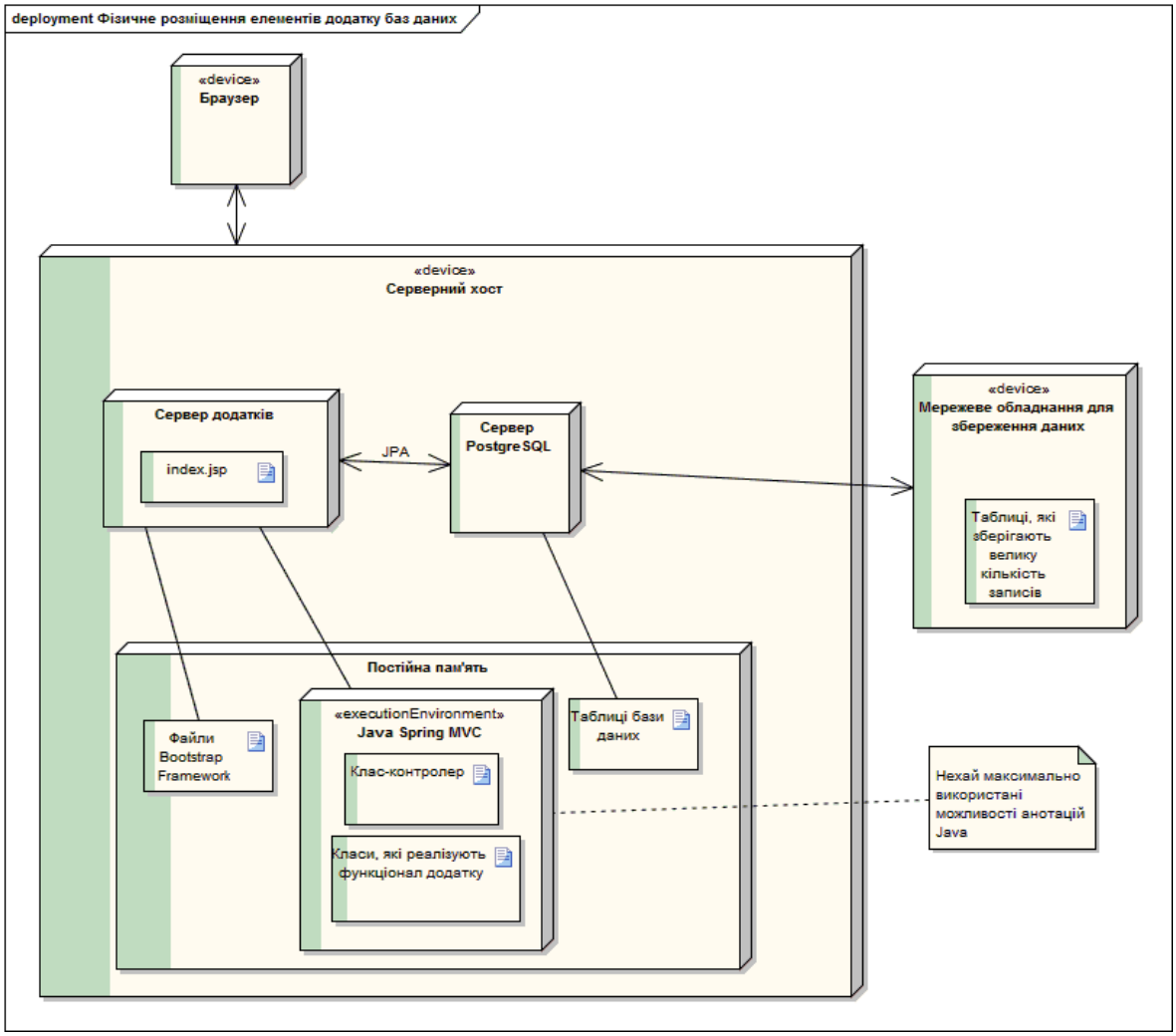


**Рис. 2.2.3.4.1. Фізичне розміщення елементів додатку баз даних**

*Кінець прикладу*

Як видно з прикладу, даний опис має невеликий обсяг, максимально лаконічний та по ньому все одно можливо зрозуміти фізичну структуру системи.

Слід зазначити, що діаграма розгортання може бути використана для опису різних рівнів абстракції системи і бути не одна в проекті (як, власне, і інші діаграми), однак для проекту, за темою прикладу очевидно, що достатньо діаграми розгортань, яка наведена вище.

Насправді діаграму розгортань можна було б використати і для представлення принципової схеми системи, однак, напевне, діаграми компонентів та комунікацій для цього підходять краще.

Різновиди діаграм можна комбінувати. Так дуже важливий клас можна винести окремо на діаграмі розгортань та дати його частковий опис. Але в

прикладі для класу-контролера не було в цьому крайньої потреби, а було необхідно показати, що він займає окреме місце в системі, як і файл index.jsp.

### 2.2.3.5.Структура баз(и) даних

Ключовим елементом звіту та курсової роботи в цілому є правильно спроектована структура бази даних. Наявність при наявності правильно оформленого звіту і робочого додатку, велика кількість суттєвих зауважень по проекту структури бази даних може послужити оцінці роботи як незадовільно виконаній[1].

При проектуванні бази необхідно виділити об’єкти предметної області і зв’язки між ними, встановити їх типи та визначити характеристики для відображення проекту бази безпосередньо в реальну баз під керуванням конкретної СУБД.

Об’єкти предметної області поділяються на атрибути та сутності.

Сутності – це об’єкти або явища матеріального чи нематеріального світів. Для простоти запам’ятовування та використання, можна розуміти сутності як іменники, які можливо виділити в предметній області.

Описати сутності повністю ніколи не можливо (це визначено ще Кантом). Але таке визначання і не потрібно, коли база проектується для конкретної предметної області. В даному випадку, для опису сутностей предметної області необхідно скористатися процедурою абстрагування. Процедура абстрагування – це процес виділення лише суттєвих для опису елементів предметної області.

**Приклад 2.2.3.5.1.**

Предметна область: склад товарів.

Початок процедури абстрагування для предметної області

Сутності:

- Товар;

[1]Власне, це визначено «Критеріями оцінювання роботи».

- Найменування товару;
- Ціна товару;
- Габарити товару;
- Вага товару;
- Реєстраційний номер товару;
- Дата прибуття товару;
- Дата списання товару;
- Дата видачі товару;
- Місце на складі;
- Номер місця на складі;
- Максимальні допустимі габарити товару для місця на складі;
- Максимальна допустима вага товару для місця на складі;
- Менеджер;
- ПІБ менеджера;
- Години роботи менеджера;
- Транспортувальник;
- ПІБ транспортувальника;
- Години роботи транспортувальника.

Нехай всі інші об’єкти є нецікавими для опису предметної області.

Початок процедури абстрагування для сутності Товар

Товар – це сукупність його найменування, ціни, габаритів, ваги реєстраційного номеру, дати прибуття, дати списання та видачі.

Кінець процедури абстрагування для сутності Товар

Початок процедури абстрагування для сутності Місце на складі

Місце на складі – це сукупність його номера, максимально допустимих габаритів для розміщення товару, максимально допустима вага для розміщення товару.

Кінець процедури абстрагування для сутності Місце на складі

Початок процедури абстрагування для сутності Менеджер

Менеджер – це сукупність його ПІБ та годин роботи

Кінець процедури абстрагування для сутності Менеджер

Початок процедури абстрагування для сутності Транспортувальник

Транспортувальник – це сукупність його ПІБ та годин роботи

Кінець процедури абстрагування для сутності Транспортувальник

Всі сутності предметної області, що залишились, є атомарними та не можуть бути описані через перелік інших виділених сутностей.

Кінець процедури абстрагування для предметної області

*Кінець прикладу*

Таким чином, з прикладу видно, що взагалі всі об'єкти предметної області не виділялися, а були визначені тільки ті які необхідно формалізувати для вирішення задачі замовника. Так, наприклад, в розглянутій процедурі абстрагування не було виділено такі сутності, як колір підлоги складу та зарплати робітників: ці об'єкти не стосуються задачі, яку поставив замовник.

Наявність зайвих об'єктів при описі структури бази може стати підставою для суттєвого зниження оцінки за курсову роботу.

Сутності, які використовуються при описі інших сутностей прийнято називати атрибутами сутностей, які описуються. Таким чином, атрибути можуть бути прості (чи атомарні) та складні (комплексні).

Зв'язки між сутностями – це формалізація асоціації проектувальника між двома об'єктами предметної області. Для простоти запам'ятовування та використання, можна розуміти зв'язки між сутностями як частини речень, в яких визначальним елементом є дієслово.

Виділяють наступні типи зв'язків між двома сутностями:

- Один-до-одного – в даному випадку кожному екземпляру однієї сутності відповідає лише один екземпляр іншої сутності і навпаки;

**Приклад 2.2.3.5.2.**

У кожного реєстраційного номеру товару може бути лише один товар і товар на складі визначається лише одним реєстраційним номером.

***Зв'язок:*** реєстраційний номер ***є атрибутом*** товару.

*Кінець прикладу*

- Один-до-багатьох – для кожного екземпляру однієї сутності може існувати багато відповідних екземплярі іншої сутності, а для кожного екземпляру другої сутності може існувати лише один відповідний екземпляр першої;

**Приклад 2.2.3.5.3.**

Кожний товар оформляє лише один менеджер, а у кожного менеджера може бути багато прийнятих товарів.

***Зв'язок:*** менеджер ***оформлює*** товари.

*Кінець прикладу*

- Багато-до-багатьох – для кожного екземпляру першої сутності може існувати багато відповідних екземплярів другої сутності і для кожного екземпляру другої сутності може існувати багато відповідних екземплярів першої сутності.

**Приклад 2.2.3.5.4.**

Кожне місце на складі може обслуговуватися багатьма транспортувальниками і кожний транспортувальник може обслуговувати багато місць на складі.

***Зв'язок:*** транспортувальники ***обслуговують*** місця на складі.

*Кінець прикладу*

Далі виділені сутності та зв'язки перетворюють у вигляд, яким може керувати конкретна СКБД як базою даних.

Часто на практиці це відбувається безпосередньо після завершення процесу абстрагування. Наприклад, відразу починають створюватися таблиці реляційної бази для конкретної СКБД.

Однак для усунення багатьох потенційних помилок додатку баз даних, які можуть виникати в процесі його експлуатації, доцільно слідувати формальній процедурі побудови структури бази даних, яка полягає у послідовному створенні наступних артефактів:

1. Концептуальної моделі даних.
2. Логічної моделі даних.

3. Фізичної моделі даних.
4. Схема розгортання бази даних

***Концептуальна модель даних***

Концептуальна модель даних – це опис асоціативних зв'язків між сутностями предметної області, які були отримані після проведення процедури її абстрагування.

Тобто, концептуальна модель бази даних була частково побудована словесно в попередніх прикладах. Для графічного представлення моделей даних використовуються спеціальні системи позначень, які мають назву нотацій. Для опису структури бази краще використовувати саме графічне представлення, побудоване по правилам обраної нотації

Наприклад, результати попередніх прикладів можуть бути зображені графічно наступним чином.

**Приклад 2.2.3.5.5.**

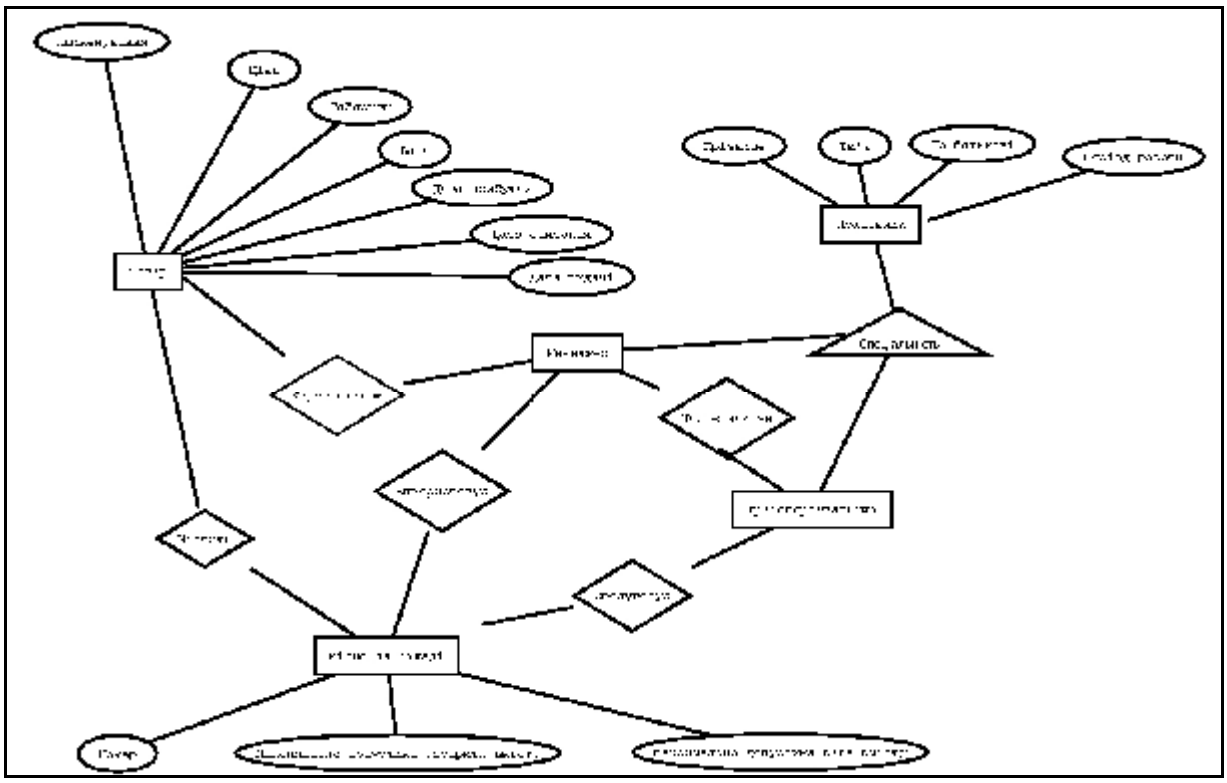

**Рис. 2.2.3.5.1. Схема концептуальної моделі бази даних складу товарів виконана по правилам нотації Чена ([6])**

*Кінець прикладу*

Як видно з малюнку, на концептуальній схемі бази даних лише визначені її сутності та зв'язки між цими сутностями без прив'язки до конкретної моделі представлення даних та фізичних особливостей розгортання бази даних.

***Логічна модель даних***

При виконання описаних раніше шляхів проектування додатку баз даних вже відомими є модель представлення даних ([4]) та конкретна СКБД, яка буде використовуватися. Ці знання складають основу для побудови логічної та фізичної моделі даних.

Логічна модель даних – це опис елементів концептуальної моделі даних в термінах та правилах конкретної моделі представлення даних.

Моделі представлення даних бувають: реляційна, ієрархічна, мережева, багатомірна, об'єктно-орієнтована, документо-орієнтована і т.д.

В логічній моделі даних сутності концептуальної моделі можуть видозмінюватися з метою уникнення можливих ситуацій порушення цілісності даних та аномалій їх оновлення. Також в логічній моделі відбувається попередня оптимізація структури для операцій з таблицями даних із бази.

*Цілісність даних*

Цілісність даних –це відповідність даних, які зберігаються в базі даних певним визначеним правилам.

В реляційній моделі представлення даних виділяють:

1. Цілісність сутностей, у відповідності до якої кожен екземпляр сутності повинен бути унікальним[2]. Для забезпечення цього виду цілісності вводиться поняття первинного ключа як обов'язкового до заповнення унікального в сукупності набору атрибутів відношення.

2. Цілісність посилань, у відповідності до якої, при наявності батьківсько-дочірніх зв'язків між відношеннями, модифікації екземплярів

[2] Інколи цілісність сутностей визначається як: всі сутності в базі повинні задовольняти визначенню поняття «Відношення».

батьківських сутностей повинні супроводжуватися їх узгодженням з залежними екземплярами дочірніх сутностей. Для забезпечення цього виду цілісності вводиться поняття зовнішнього ключа, як атрибуту сутності який належить також і батьківській сутності. При спробі змін екземплярів батьківської сутності, типова СКБД виконує певний визначений набір дій, напрямлений на запобігання порушення даного виду цілісності.

На фізичному рівні моделювання слід визначити ще два обмеження цілісності:

3. Доменна цілісність, у відповідності до якої, значення атрибутів екземплярів повинні бути в межах доменів цих атрибутів. Доменна цілісність забезпечується визначенням доменів доступних для обраної СКБД.

4. Користувацька цілісність, у відповідності до якої екземпляри сутностей повинні задовольняти додаткові правила користувача. Правила користувача реалізуються за допомогою користувацького інтерфейсу та процедур і тригерів СКБД.

*Аномалії оновлення даних*

Аномалії оновлення даних – це ситуації надлишковості, втрати даних, або наявності завад керування СКБД внаслідок виконання операцій INSERT, DELETE, UPDATE, якщо розглядається реляційна модель даних.

В реляційній моделі даних, для уникнення ситуацій аномалій оновлення даних, при проектування логічної схеми, використовується процедура нормалізації, яка полягає у декомпозиції сутностей для того, щоб вони задовольняли певні визначені правила.

Ці правила визначаються, так званими, нормальними формами. Виділяють: першу, другу, третю, третю Бойса-Кодда, четверту та п’яту нормальні форми.

У [3] дані наступні визначення нормальним формам та допоміжні до них.

Функціональна залежність. Нехай $r$ є відношенням, а $X$ та $Y$ - довільними підмножинами множин атрибутів відношення $r$. Тоді $Y$ функціонально залежить від $X$ ($X \to Y$) тоді та тільки тоді, коли кожне значення множини $X$ відношення $r$ зв’язано рівно з одним значенням

множини $Y$ даного відношення. $X$ називається детермінантом функціональної залежності, а $Y$ - залежною частиною.

Перша нормальна форма. Змінна-відношення знаходиться в першій нормальній формі тоді і тільки тоді, коли в будь-якому прийнятному значенні цієї змінної-відношення кожний її кортеж містить тільки одне значення для кожного з атрибутів[3].

Неприводима функціональна залежність. Множина функціональних залежностей називається неприводимою тоді та тільки тоді, коли їй притаманні наступні властивості:

- Залежна частина кожної функціональної залежності множини функціональних залежностей відношення містить лише один атрибут.
- Детермінант кожної функціональної залежності з цієї множини є неприводимим, тобто жоден атрибут з детермінанту не може бути виключений без зміни замикання множини функціональних залежностей відношення. В цьому випадку функціональна залежність називається неприводимою зліва.
- Жодна функціональна залежність з множини функціональних залежностей відношення не може бути виключена з неї без зміни її замикання.

Друга нормальна форма. Змінна-відношення знаходиться в другій нормальній формі тоді та тільки тоді, коли вона знаходиться в першій нормальній формі і кожний неключовий атрибут неприводимо залежить від її первинного ключа.

Третя нормальна форма. Змінна-відношення знаходиться в третій нормальній формі тоді та тільки тоді, коли вона знаходиться в другій нормальній формі і кожний неключовий атрибут нетранзитивно залежить від її первинного ключа.

[3] Інколи першу нормальну форму визначають так: Таблиця бази знаходиться в першій нормальній формі, якщо відповідне відношення задовольняє визначенню відношенню по Кодду.

Багатозначна функціональна залежність. Нехай $A$, $B$ та $C$ є довільними підмножинами множини атрибутів змінної-відношення $R$. Підмножина $B$ багатозначно залежить від підмножини $A$ ($A$®® $B$) тоді та тільки тоді, коли множина значень $B$, що відповідає парі $(A,C)$ змінної-відношення $R$, залежить від $A$, але не залежить від $C$.

Третя нормальна форма Бойса-Кодда. Змінна-відношення знаходиться в третій нормальній формі Бойса-Кодда тоді та тільки тоді, коли детермінанти всіх її функціональних залежностей є потенційними ключами.

Четверта нормальна форма. Змінна-відношення $R$ знаходиться в четвертій нормальній формі тоді та тільки тоді, коли у випадку існування таких підмножин $A$ і $B$ атрибутів цієї змінної-відношення $R$, для яких виконується нетривіальна багатозначна залежність $A$®® $B$, всі атрибути змінної відношення $R$ також залежать від атрибута $A$.

Залежність з'єднання. Нехай $R$ є змінною-відношенням, а $A,B,...,Z$ - довільними підмножинами множини її атрибутів. Змінна-відношення $R$ задовольняє залежності з'єднання ($*\{A,B,...,Z\}$) тоді та тільки тоді, коли будь-яке допустиме значення змінної-відношення $R$ еквівалентно з'єднанню її проекцій по підмножинам атрибутів $A,B,..,Z$.

П'ята нормальна форма. Змінна-відношення знаходиться в п'ятій нормальній формі, тоді та тільки тоді, коли кожна нетривіальна залежність з'єднання в даній змінній-відношенні визначається її потенційними ключами.

Більш детальну інформацію про нормальні форми можна знайти у [3].

*Оптимізація структури під операції з таблицями даних*

Декомпозиція сутностей може призвести до створення їх великої кількості. Якщо тепер виконувати операції вибірки, вставки, оновлення та видалення даних над таблицями бази, то можуть виникнути зайві затрати часу на аналіз взаємозв'язків між таблицями в процесі виконання операцій. Для уникнення цієї ситуації, процес декомпозиції необхідно узгоджувати з вимогами до швидкодії названих операцій. І інколи виникають ситуації, коли існує висока

вірогідність виникнення аномалій оновлення даних, але з метою підвищення швидкодії виконання операцій, цей процес зупиняється.

*Приклад побудови логічної схеми бази даних*

У звіті з курсової роботи необхідно привести вже готову логічну схему і процес нормалізації показувати необхідності немає.

В цілому, логічна схема повинна бути описана подібно до випадку, розглянутому у прикладі.

**Приклад 2.2.3.5.6.**

Логічна модель бази даних складу товарів представлена на Рис. Рис. 2.2.3.5.2.

*Зауваження*. З Рис. 2.2.3.5.2. видно, що зменшення масштабу його вмісту, зробить схему нечитабельною. Важлива умова при оформленні звіту з курсової роботи: зміст логічної схеми повинен бути прочитаний без зусиль. Якщо схема виявляється більшою за наведену у прикладі, то необхідно для її розміщення використати декілька Рисунків і, відповідно, – сторінок. Схеми з важко читабельним змістом будуть вважатися повністю не зданими з відповідними наслідками при оцінюванні роботи.

Як видно з рисунку, в логічній схемі, яка все ж створювалася на основі концептуальної, значно більше сутностей.

Було виділено наступні сутності:

1. Товар. Ця сутність описує основні характеристики товару.
2. ЦінаТовару. Оскільки ціна товару може змінюватися під час його зберігання, а планування використання операції UPDATE є вкрай невдалим рішенням при проектуванні, було прийнято рішення про створення цієї сутності.
3. СтанТовару. Ця сутність дозволяє зберігати історію змін стану товару (прийнято, списано, видано) шляхом збереження його станів та часу виставлення стану.
4. Працівник. Оскільки ролі менеджера і транспортувальника може виконувати одна людина, чи в процесі роботи на склад вона може змінювати

свою професію, то доцільно ввести загальну інформацію про персону в одну таблицю, а службову – в інші. Сутність Працівник, містить дані про громадянина, який працює на склад.

5. СпеціальністьПрацівника. Ця сутність дозволяє створювати штатні одиниці складу. В даній таблиці один Працівник може мати декілька записів, що відображає його кар'єру або сумісництво.

6. ГрафікСпеціальностіПрацівника. В даній сутності для кожної штатної одиниці створюються одиниці проміжків часу, в межах яких вона працює.

7. СтанЕлементуГрафікуСпеціальностіПрацівника. В даній сутності кожній одиниці процесу роботи працівника виставляється її статус (відкрита, лікарняний, відпустка, закрита) та час встановлення статусу.

8. МісцеНаСкладі. Сутність охоплює основні фізичні характеристики кожного місця на складі.

9. ТоварНаМісці. Дана сутність пов'язує товар з місцем його розташування на складі. Також вона дозволяє визначити менеджера та транспортувальника, які обслуговували операцію його розміщення.

10. БлокуванняМісцяНаСкладі. Дана сутність дозволяє в базі зобразити факт блокування-розблокування місця на складі з технічних причин, не пов'язаних з замовленим процесом.

*Зауваження*. Для прикладу далі документується сутність МісцеНаСкладі, але в курсовій роботі необхідно задокументувати всі спроектовані сутності. *Кінець зауваження*.

Місце на складі має наступні атрибути:

- номер – номер, за яким ідентифікується місце. Для цього атрибуту необхідно використати цілочисельний тип даних у фізичній моделі.

- максимальнаДовжина – максимальна довжина товару, який може вмістити дане місце. Для цього атрибуту необхідно використати дійсний тип даних.

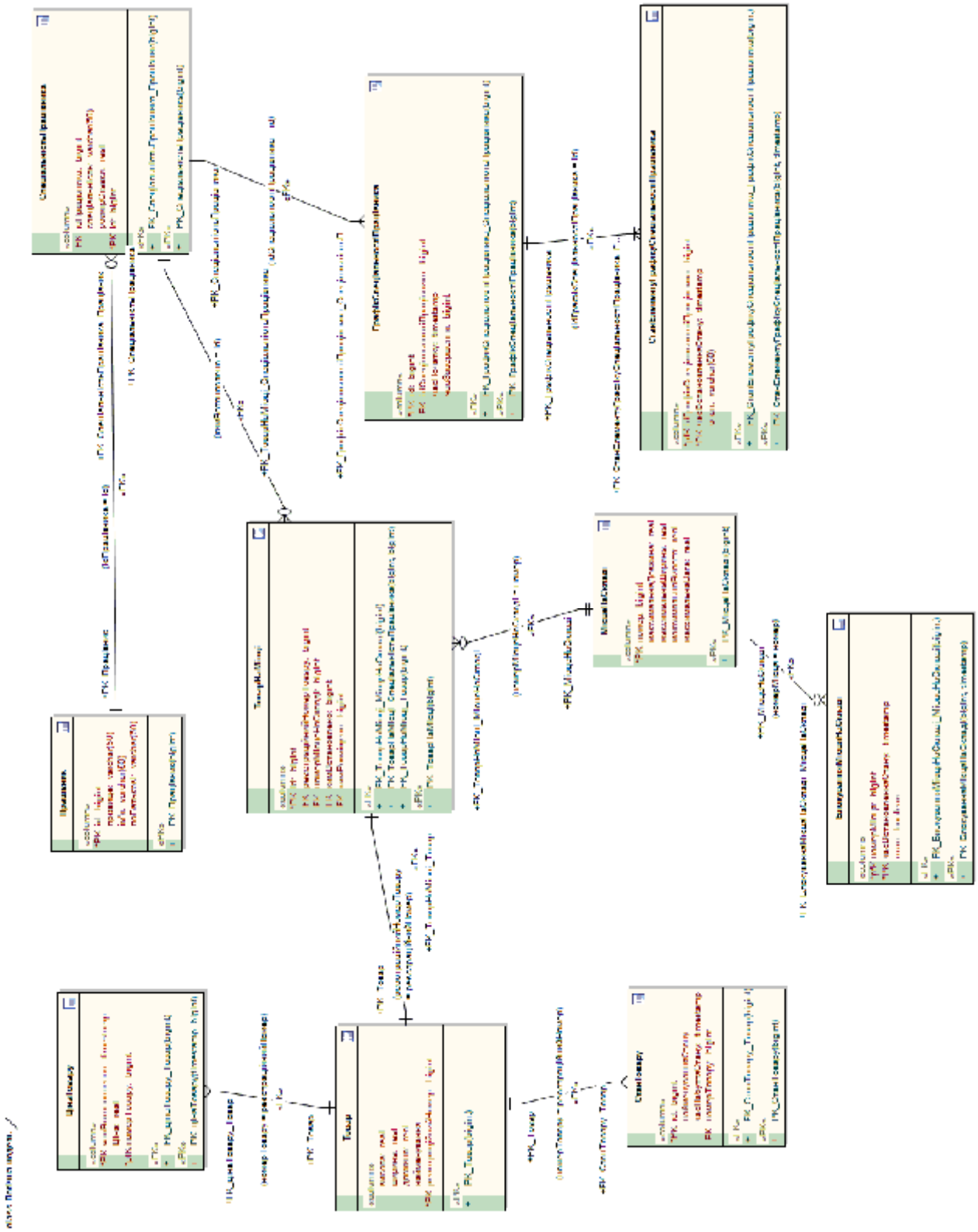

**Рис. 2.2.3.5.2. Логічна схема бази даних складу товарів, виконана в нотації «Вороняча лапка» (Crow’s Foot)**

- максимальнаШирина – максимальна ширина товару, який може вмістити дане місце. Для цього атрибуту необхідно використати дійсний тип даних.

- максимальнаВисота – максимальна висота товару, який може вмістити дане місце. Для цього атрибуту необхідно використати дійсний тип даних.

- максимальнаВага – максимальна вага товару, який може вмістити дане місце. Для цього атрибуту необхідно використати дійсний тип даних.

В якості первинного ключа сутності виступає атрибут – номер.

*Кінець прикладу*

***Фізична модель даних***

Фізична модель даних використовується для проектування схеми бази даних з урахуванням властивостей СКБД, під керуванням якої вона буде знаходитися.

В ній необхідно визначити структуру реальних колонок таблиць, які відповідають сутностям та опрацювати представлення, тригери, процедури бази та її допоміжні конструкції.

**Приклад 2.2.3.5.7.**

На Рис. 2.2.3.5.3 Представлена фізична схема бази даних складу товарів.

Для спрощення пошуку вільних місць на складі, було створено представлення FreePalcesInStore, яке в результаті повертає множину кортежів з тими самими атрибутами, що і таблиця PlaceInStore, але, в результаті виконання відповідного SQL-запиту, дана множина буде містити лише незайняті та незаблоковані місця на складі.

*Зауваження*. В фізичній моделі був зроблений переклад назв об’єктів бази, оскільки навіть якщо СКБД та технології програмування, під які проектується база, здатні на даний момент ефективно працювати з Кирилецею, не виключено, що в майбутньому може бути задіяна технологія не здатна на це, що займе додатковий час та гроші для адаптації нової технології в існуючу систему. *Кінець зауваження.*

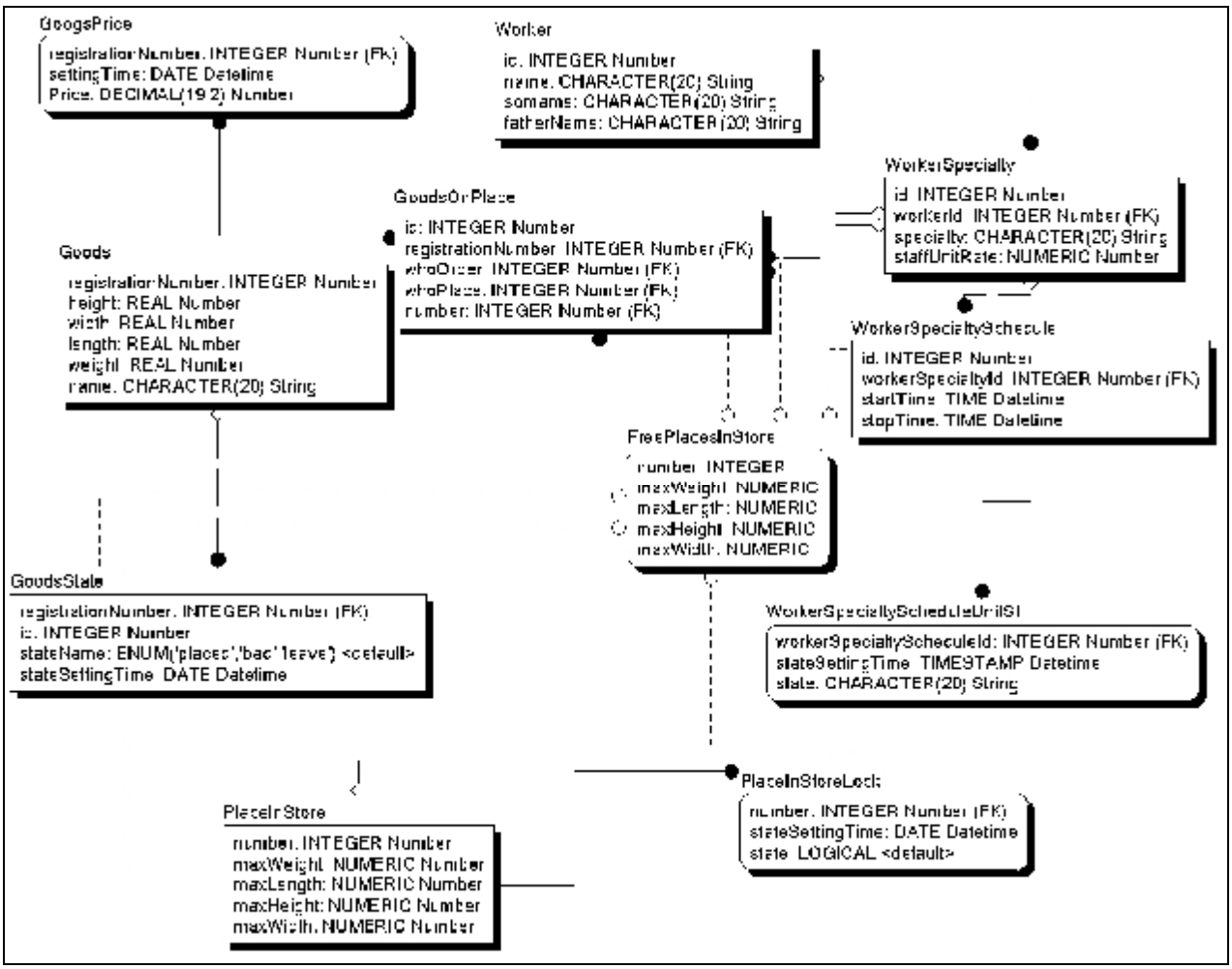


**Рис. 2.2.3.5.3. Фізична схема бази даних складу товарів, виконана в нотації IDEF 1x**

*Зауваження*. В звіті з курсової роботи потрібно коротко (не більше одного речення) обґрунтувати вибір типу даних для кожного атрибуту. *Кінець зауваження.*

*Кінець прикладу*

***Схема розгортання бази даних***

На даному етапі слід повернутися до діаграми розгортання додатку бази даних та, відкинувши її елементи, які не стосуються СКБД та бази даних розмістити таблиці бази по хостах.

**Приклад 2.2.3.5.8.**

Схема розгортання бази даних представлена на Рис. 2.2.3.5.8.

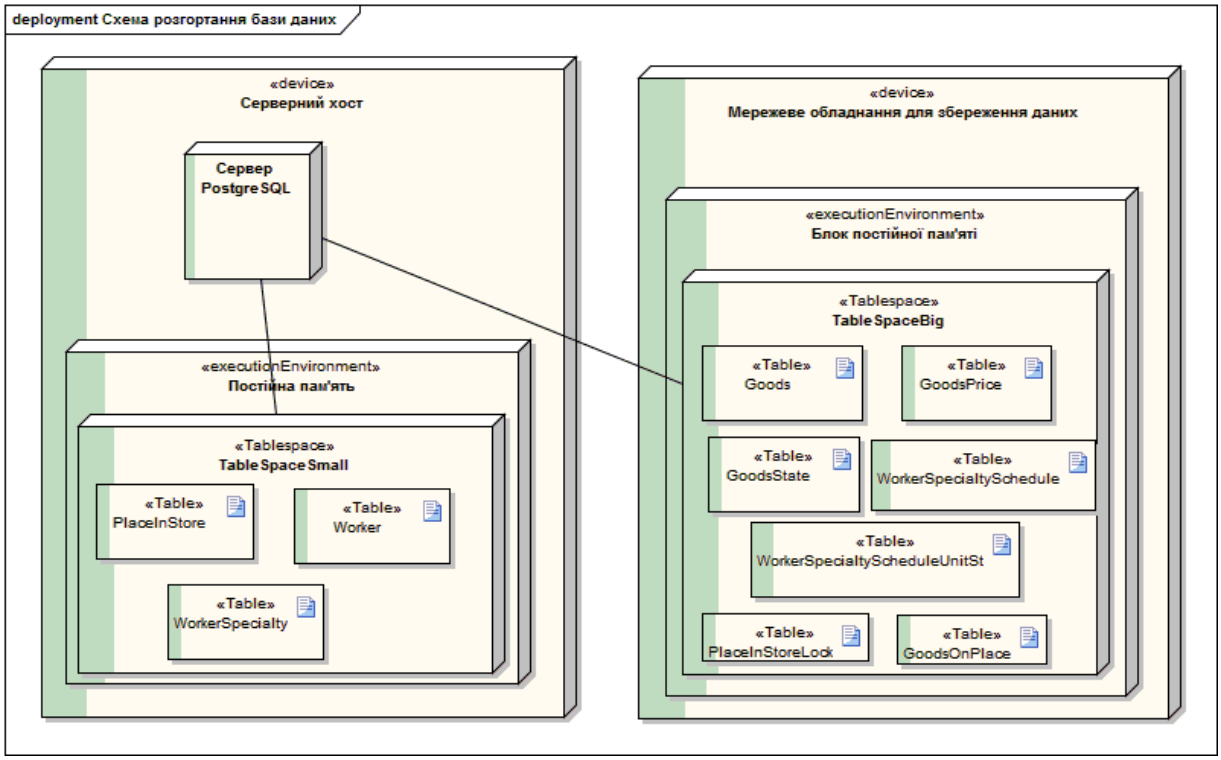


**Рис. 2.2.3.5.4. Діаграма розгортання бази даних**

Таблиця PlaceInStore розміщена на серверному хості, оскільки вона повинна мати невеликий розмір та її можна використати для швидкого пошуку місць на складі, що відповідають товару без урахування їх стану.

Таблиця Worker розміщена на серверному хості, оскільки вона теж має відносно невеликий розмір.

Таблиця WorkerSpecialty потенційно повинна бути більшою в декілька разів за Worker, однак також мати відносно невеликий розмір. Ця таблиця дозволяює швидко зменшити множину менеджерів та транспортувальників, серед яких необхідно вибрати виконавців розміщення товару на складі.

Таблиці Goods, GoodsPrice, GoodsState, WorkerSpecialtySchedule, WorkerSpecialtyScheduleUnitSt, PlaceInStoreLock, GoodsOnPlace потенційно можуть мати дуже велику кількість записів, тому при проектуванні прийнято рішення про розміщення цих таблиць на мережевому обладнанні, яке спеціалізується на збереженні даних.

*Зауваження*. При проектуванні діаграми розгортання було зазначено, що мережеве обладнання для збереження даних має спеціальні засоби захисту інформації. З цієї причини, якщо замовник висуне вимогу захищенності даних

бази, може бути доцільно розмістити всі її бази на цьому обладнанні і нічого не залишати на хості. При рівноцінності вимог до швидкодії операцій із СКБД та захищенності даних необхідно знаходити розумний компроміс при розміщенні таблиць. *Кінець зауваження.*

*Кінець прикладу*

***Правила та рекомендації для виконання елементу звіту***

При описі структури бази даних припустимо поєднати її концептуальний та логічний опис.

Для опису логічних та фізичних схем можна використовувати будь-яку нотацію, однак завжди намагаючись задовільнити два критерії: доступність викладення та мінімальний об’єм тексту.

Для побудови схем можна використовувати будь-який case-засіб: ERWin, Enterprise Architech Data Modelling Diagrams, Dia, Edraw Max, Microsoft Visio і т.д.

Формальним критерієм якості виконаного проектування бази даних курсової роботи є: знаходження всіх таблиць бази в четвертій нормальній формі, прийняті всі міри для збереження обмежень цілістності та, як вже зазначалося, у логічній схемі повинно бути не менше десяти сутностей.

Однак, окрім формальних критеріїв, до схем бази можуть бути зроблені додаткові зауваження по їх змісту.

При наявності у схемах штучно утворених сутностей та таблиць, оцінка за курсову роботу буде знижена.

### 2.2.3.6.Діаграма класів додатку баз даних

Питання про те, що необхідно проектувати першим: таблиці бази даних, чи класи додатку є спірним.

Безумовно спроектовані класи додатку дозволять уникнути зайвої формалізації понять предметної області при створенні таблиць бази даних. З іншого боку, якщо в класах не задіяні деякі поняття предметної області, які дуже бажано б було використати при побудові логічної схеми бази даних, то скоріш за все додаток баз даних у майбутньому доведеться допрацьовувати через його занадто вузьку направленість.

Взагалі кажучи, логічну схему бази даних можна зображувати на одній діаграмі з класами додатку баз даних, взаємно поєднуючи програмні сутності та сутності даних зв'язками. Якщо уважно придивитися до інструментарію окремих case-засобів для побудови діаграм (наприклад, Enterprise Architect), то можна побачити, що ця концепція там реалізована.

В принципі, у звіті з курсової роботи можна поєднати побудови схеми бази даних з структурою класів її програмного додатку.

Але у даних вказівках все ж буде розглянуто ці два процеси окремо.

В діаграмі класів додатку баз даних повинні бути ієрархії. Відсутність ієрархій свідчить про невисоку ефективність проекту структури класів, чи занадто просту предметну область додатку баз даних.

Оформлювати же представлення схеми класів додатку необхідно подібно до того, як наведено в наступному прикладі.

**Приклад 2.2.3.6.1.**

Діаграма класів для додатку баз даних складу товарів зображена на наступному Рис. 2.2.3.6.1.

*Зауваження.* При описі схеми класів додатку баз даних, необхідно визначитися які сутності підлягають опису та на якому рівні абстракції. Описувати всі класи додатку, як вже відзначалося раніше, не потрібно.

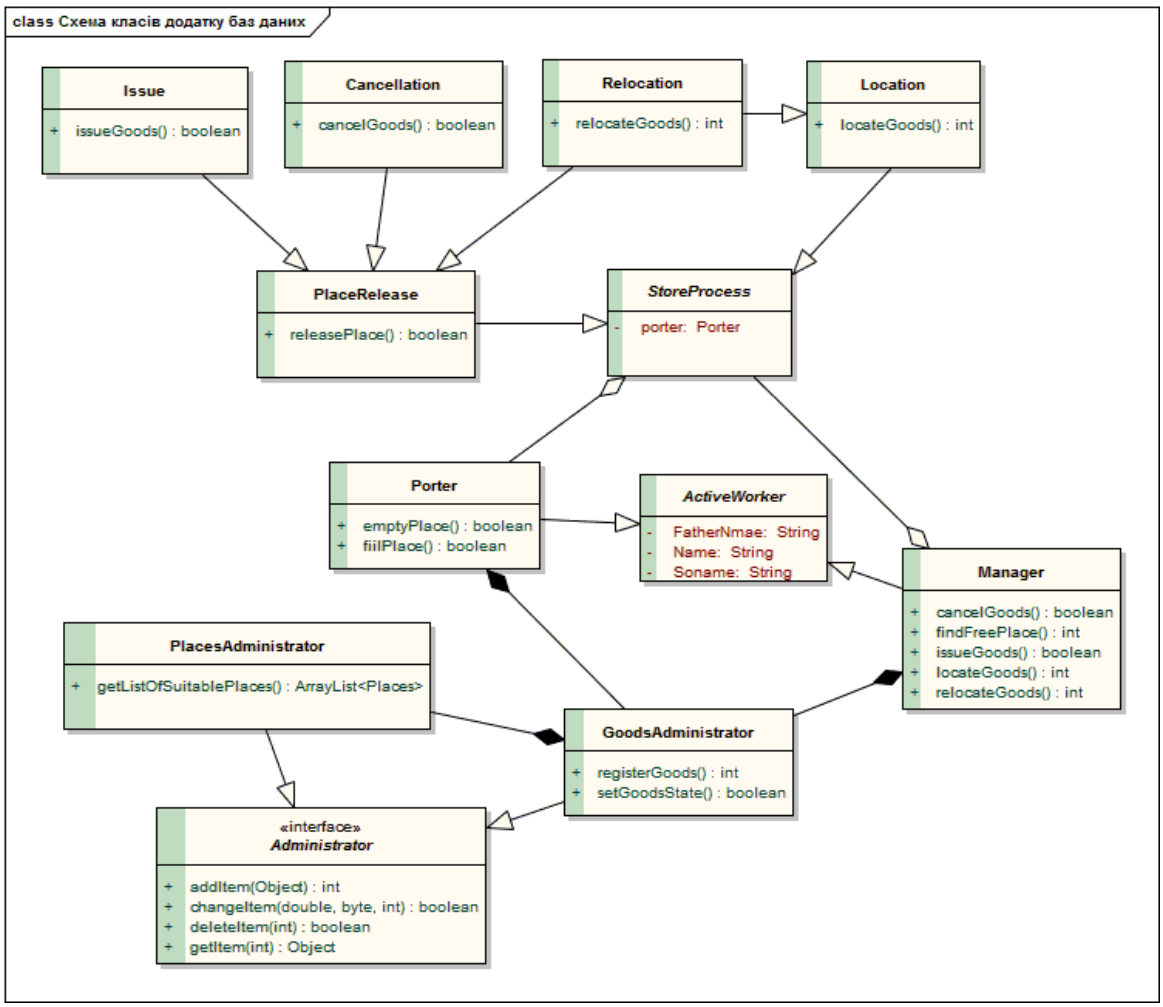


**Рис. 2.2.3.6.1. Діаграма класів для складських процесів, пов'язаних з товаром**

Це може призвести до того, що при читанні діаграми буде важко виділяти елементи предмету опису. Так на вище наведеному рисунку не описується диспетчерний сервлет, DAO класи та сутністні класи (хоча два таки наведені, але це дозволяє покращити викладення предмету опису). Окрім того, на рисунку видно, що класи описані не детально. На класах виділені лише елементи (об'єкти та методи), які або обґрунтовують їх окреме існування, або визначають важливі елементи предмету, який описується. Якщо ж в курсовій можна виділити декілька предметів, які гідні опису, то можна навести декілька діаграм класів.

Отже, власне, сам опис:

При необхідності проведення нового процесу з товаром створюється об'єкт класу Менеджер, який керує проведенням відповідного процесу.

Оскільки всі процеси виконуються певним транспортувальником, то створюється і об’єкт класу, який є дочірнім до абстрактного класу StoreProcess, одним з атрибутів якого є об’єкт класу Porter.

При виконанні операцій об’єктами класів Manager та Porter задіюється фабрика, яка повертає посилання на один і той же об’єкт класу GoodsAdministrator.

Об’єкт GoodsAdministrator, задіюючи фабрику класів, отримує єдиний об’єкт класу PlacesAdministrator, який використовується для управлінням місцями на складі.

В своїй роботі об’єкти GoodsAdministrator та PlacesAdministrator задіюють сутністні класи, які за допомогою технології JPA дозволяють отримувати записи таблиць бази.

Клас StoreProcess є, як вже було сказано, абстрактним та для проведення реальних процесів використовуються його наслідники:

- Location – для розміщення товару на складі;
- Relocation – для переміщення товару на складі;
- Cancellation – для списання товару;
- Issue – для видачі товару.

Клас PlaceRelease безпосередньо не використовується, а визначає метод releasePlace(), який дозволяє визначити в базі факт звільнення конкретного місця на складі.

*Зауваження*. Даний опис можна було б супроводжувати діаграмами взаємодії. *Кінець зауваження.*

*Кінець прикладу*

Далі, кожний клас необхідно детально описати.

**Приклад 2.2.3.6.2.**

Administrator є інтерфейсом, який реалізується класами GoodsAdministrator та PlacesAdministrator та він містить наступні елементи:

- int addItem(Object oItem); - декларація, яка визначає необхідність реалізації в дочірніх класах методу додавання елементу в базу. Атрибут oItem – це об’єкт, який підлягає додаванню. Метод повертає реєстраційний номер елементу.

- boolean changeItem(int iNum, byte bType, double dValue); - декларація, яка визначає необхідність реалізації в дочірніх класах методу зміну стану елементу в базі. iNum – це ідектифікаційний номер елемента в системі, bType – це тип зміни, а dValue – може містити значення, яке необхідно при певних типах змін. У разі успіху метод повертає true та false – в разі неуспішного завершення.

- boolean deleteItem(int iNum); - декларація, яка визначає необхідність реалізації в дочірніх класах методу видалення елементу з бази. iNum визначає номер елемента, що необхідно видалити. У разі успіху метод повертає true та false – в разі неуспішного завершення.

- Object getItem(int iNum)- декларація, яка визначає необхідність реалізації в дочірніх класах методу повернення елементу за ознакою. iNum визначає номер елемента, що необхідно повернути. Метод повертає або запитаний об’єкт, або null.

*Кінець прикладу*

Описи, подібні до наведеного в прикладу, необхідно дати для всіх виділених класів.

Оскільки основна структура програми зображена на діаграмі класів та була описана в попередніх діаграмах, додавати до звіту з курсової роботи лістінги артефактів додатку баз даних не потрібно.

### 2.2.3.7. Обґрунтування доцільності та ефективності запитів до СКБД в додатку баз даних

Для обґрунтування доцільності запитів достатньо визначити задачу, яку вони вирішують в додатку баз даних.

Обґрунтування оптимальності запиту вимагає математичного доведення цього факту. У звіті з курсової роботи таке доведення роботи не потрібно.

Ефективність запиту – це його здатність вирішити поставлену задачу. Окрім того, одну і ту ж задачу можна вирішити різними шляхами: одні будуть кращі з точки зору швидкодії виконання операцій, інші – гірші. Тому під ефективністю запиту у звіті з курсової роботи буде розумітися його переважна ефективність перед іншими можливими рішеннями задачі. Знайти ефективне рішення можливо, порівнюючі між собою можливі альтернативи.

Алгоритм обґрунтування доцільності та ефективності запиту до СКБД складається з наступної послідовності кроків:

1. Формулювання задачі, яку вирішує запит.
2. Перелік таблиць, які будуть задіяні у запиті та короткий опис мети їх використання.
3. Опис даних таблиць на мові DDL.
4. Код процедури заповнення їх змістом для тестування.
5. Текст SQL-запитів, які порівнюються між собою.
6. З використанням операторів EXPLAIN та ANALYZE отримати чисельні характеристика запитів.
7. Зробити висновки.

**Приклад 2.2.3.7.1.**

1. Задача: зробити вибірку всіх транспортувальників, які на даний момент не перебувають у відпустці.

*Зауваження*. На цьому обґрунтування доцільності завершено. *Кінець зауваження.*

2. Перелік таблиць, які будуть задіяні у запиті:

Очевидно, що в процесі запиту будь задіяні таблиці WorkerSpecialty, WorkerSpecialtySchedule та WorkerSpecialtyScheduleUnitSt

З таблиці WorkerSpecialty можна буде визначити номери всіх транспортувальників.

З таблиці WorkerSpecialtySchedule можна буде визначити їх розклад роботи.

З таблиці WorkerSpecialtyScheduleUnitSt можна буде визначити актуальність кожної визначеного з таблиці WorkerSpecialtySchedule одиниці розкладу.

3. Опис задіяних у запитах таблиць на мові DDL.

Таблиця WorkerSpecialty

```
create type profession as enum('manager', 'porter');
create table
    WorkerSpecialty(
                    id bigserial primary key,
                    workerId bigint,
                    specialty profession,
                    stafUnitRate real,
                    foreign key(workerId) references Worker(id));
```

Таблиця WorkerSpecialtySchedule

```
create table
   WorkerSpecialtySchedule(
                        id bigserial primary key,
                        workerSpecialtyId bigint,
                        startTime time without time zone,
                        stopTime time without time zone,
                        check (startTime < stopTime),
                        foreign key(workerSpecialtyId)
                                references WorkerSpecialty(id));
```

WorkerSpecialtyScheduleUnitSt

```
create type UnitState as enum ('active','vacation','inactive');
create table
        WorkerSpecialtyScheduleUnitSt(
                                workerSpecialtyScheduleId bigint,
```

```
                                    stateSettingTime timestamp without time zone,
                                     state unitState,
                                    foreign key(workerSpecialtyScheduleId)
                                            references WorkerSpecialtySchedule(id),
                                    primary key(
                                                    workerSpecialtyScheduleId,
                                                    stateSettingTime)
                                     );
```

4. Код процедур заповнення таблиць змістом для тестування.

*Головна процедура заповнення досліджуваних таблиць змістом.*

```
create or replace function filler() returns void as
$$
declare
i bigint;
j bigint;
begin
        truncate worker cascade;
        alter sequence Worker_id_seq restart with 1;
        alter sequence WorkerSpecialty_id_seq restart with 1;
        alter sequence WorkerSpecialtySchedule_id_seq restart with 1;

        for i in 1..1000 loop
                insert into Worker(name, soname, fathername)
                        values(substr(md5(random()::text),1,20),
                                substr(md5(random()::text),1,20),
                                substr(md5(random()::text),1,20)
                               );
                insert into WorkerSpecialty(workerId,specialty,stafUnitRate)
                                        values(i,'porter',0.5);
```

```
            perform fillerChilds();



            j := trunc(random()*2);
            if (j = 1) then
                  insert into workerSpecialty(workerId,specialty,stafUnitRate)
                                          values(i,'manager',0.5);
                  perform fillerChilds();
            end if;

      end loop;
end;
$$ language plpgsql;
```

*Допоміжна процедура, яку використовує filler()*

```
create or replace function fillerChilds() returns void as
$$
declare
k bigint;
cv bigint;
tmp int;
tmWSS time;
cvWSS bigint;
tmpWSS int;
begin
      cv:= currval('WorkerSpecialty_id_seq');
      tmp:= trunc(random()*5 + 1);
      for k in 1..tmp loop
            tmWSS := current_time - time '04:00:00' + random()* (time '08:00:00');
            insert into WorkerSpecialtySchedule(workerSpecialtyId,
```

```
                                        startTime,
                                         stopTime)
                        values(
                                cv,
                                tmWSS,
                                tmWSS + '03:30:00'
                        );
        cvWSS := currval('WorkerSpecialtySchedule_id_seq');
        tmpWSS := trunc(random()*3 + 1);

        insert into WorkerSpecialtyScheduleUnitSt(workerSpecialtyScheduleId,
                                                  stateSettingTime,
                                                  state
                                                 )
                                values
                                (
                                        cvWSS,
                                        localtimestamp - time '00:30:00',
                                        'active'
                                );
        if(tmpWSS > 1) then
        insert into WorkerSpecialtyScheduleUnitSt(
                                                  workerSpecialtyScheduleId,
                                                  stateSettingTime,
                                                 state
                                                 )
                                        values
                                        (
                                                cvWSS,
                                                localtimestamp
```

```
                                        - time '00:20:00',
                                        'vacation'
                                    );
        end if;
        if(tmpWSS > 2) then
            insert into WorkerSpecialtyScheduleUnitSt(
                                        workerSpecialtyScheduleId,
                                            stateSettingTime,
                                            state
                                        )
                                    values
                                    (
                                        cvWSS,
                                        localtimestamp
                                        - time '00:10:00',
                                        'active'
                                    );
            insert into WorkerSpecialtyScheduleUnitSt(
                                        workerSpecialtyScheduleId,
                                            stateSettingTime,
                                            state
                                        )
                                    values
                                    (
                                        cvWSS,
                                        localtimestamp
                                        - time '00:09:00',
                                        'inactive'
                                    );
        end if;
```

```
    end loop;

end;
$$language plpgsql;
```

5. Текст SQL-запитів, які порівнюються.

*Запит 1:*

```
select WorkerSpecialty.id
from WorkerSpecialty,
     WorkerSpecialtySchedule,
     WorkerSpecialtyScheduleUnitSt tbl
where
(WorkerSpecialty.id = WorkerSpecialtySchedule.workerSpecialtyId)
and
(WorkerSpecialtySchedule.id
          = tbl. workerSpecialtyScheduleId)
and
(current_time between WorkerSpecialtySchedule.startTime
                      and WorkerSpecialtySchedule.stopTime)
and
(tbl.state = 'active')
and
(
 tbl.stateSettingTIme
 =
 (
        select max(stateSettingTIme)
        from WorkerSpecialtyScheduleUnitSt
        where
```

```
            workerSpecialtyScheduleId =
            tbl. workerSpecialtyScheduleId )
    );
```

*Запит 2:*

```
select WorkerSpecialty.id
from WorkerSpecialty,
    WorkerSpecialtySchedule,
   (
        select tblh.workerSpecialtyScheduleId,
               tblh.stateSettingTime,
              WorkerSpecialtyScheduleUnitSt.state
         from WorkerSpecialtyScheduleUnitSt,
                    (select
                          workerSpecialtyScheduleId,
                          max(stateSettingTIme) as stateSettingTIme
                     from WorkerSpecialtyScheduleUnitSt
                     group by workerSpecialtyScheduleId
                    ) tblh
         where
        (
            WorkerSpecialtyScheduleUnitSt. workerSpecialtyScheduleId
            = tblh.workerSpecialtyScheduleId
        )
            and
            (
                  WorkerSpecialtyScheduleUnitSt.stateSettingTIme =
                  tblh.stateSettingTIme
            )
    ) tbl
```

```
where
        (WorkerSpecialty.id = WorkerSpecialtySchedule.workerSpecialtyId)
and
        (WorkerSpecialtySchedule.id = tbl.workerSpecialtyScheduleId)
and
        (current_time between WorkerSpecialtySchedule.startTime
                                and WorkerSpecialtySchedule.stopTime)
and
       (tbl.state = ‘active’);
```

6. Аналіз досліджуваних запитів

Перед виконанням аналізу запитів, до всіх досліджуваних таблиць застосовується оператор ANALYZE для поміщення статистики про них до таблиці pg_statistic. При виконанні запитів, планувальник буде використовувати цю інформацію для оптимізації своєї роботи.

Отже,

**ANALYZE** WorkerSpecialty;

**ANALYZE** WorkerSpecialtySchedule;

**ANALYZE** WorkerSpecialtyScheduleUnitSt;

*Аналіз запиту 1*

**EXPLAIN (Format yaml)** ***{тут скопійовано та вставлено текст першого запиту, але для зменшення об’єму даних методичних вказівок, ще раз цей код не наводиться};***

Результат:

```
- Plan:
                        +
    Node Type: "Nested Loop"
                        +
    Join Type: "Inner"
                        +
```

```
Startup Cost: 0.56
                    +
Total Cost: 5994.82
                    +
Plan Rows: 3
                    +
Plan Width: 8
```

*Зауваження.* Інформацію про дочірні плани виводити не потрібно.

*Аналіз запиту 2.*

**EXPLAIN (Format yaml)** ***{Запит 2};***

Результат:

```
- Plan:
                              +
    Node Type: "Nested Loop"
                              +
    Join Type: "Inner"
                              +
    Startup Cost: 400.88
                              +
    Total Cost: 724.75
                              +
    Plan Rows: 3
                              +
    Plan Width: 8
```

7. Висновки

Оскільки показник Total Cost другого запиту значно менше за даний показник для першого запиту, доцільно реалізувати в програмному додатку саме другий запит.

*Зауваження*. Також для аналізу можна було б використати запит:

**EXPLAIN (format yaml, analyze true, timing true)** ***{Запит}***

Тоді висновки можна було б робити не тільки на основі кількості та довжини рядків, а і на підставі часу, який затрачається на вибірки.

Власне, для наведених у прикладі запитів були отримані наступні значення:

Запит 1: 146,068

Запит 2: 118,066

Тобто, час відрізняється не сильно, але в перспективі (при збільшенні кількості записів в базі) ця різниця може стати суттєвою.

Очевидно, що аналіз статичних характеристик плану у порівнянні з аналізом кількості часу, який затрачається на виконання плану, є більш наочним. Тому, у курсовій можна виконувати аналіз лише на основі значення показника Total Cost. *Кінець зауваження.*

*Зауваження.* З прикладу постає питання про можливість виключення таблиці WorkerSpecialtyScheduleUnitSt з бази та, при оновлені статусу штатної одиниці, використовувати оператор UPDATE стосовно таблиці WorkerSpecialtySchedule для поля status, яке, в такому разі, необхідно включити до неї.

Однак, якщо в системі, яка проектується, планується можливе розширення на підсистеми бухгалтерії та відділу кадрів, то можливість виключення таблиці WorkerSpecialtyScheduleUnitSt відсутня, оскільки за її використання є можливим проведення ряду процесів даних підсистем. *Кінець зауваження.*

*Зауваження*. Також, для штатної одиниці, окрім єдиних годин роботи можна ввести різні години для різних днів тижня (таке на практиці буває не часто, але буває і, тому в реальній системі необхідно було б передбачити цю можливість). Для цього потрібно було б виконати наступні два додаткові кроки:

1. Створити новий тип с символьними представленнями днів тижня:

```
create type wdays as enum('Sun', 'Mon',’Tue’,’Wed’,’Thu’,’Fri’,’Sat’);
```

Або, просто взяти для майбутнього поля цілий тип даних та обмежити його значення від 1 до 7. Автору цей варіант здається більш прийнятним, оскільки дані позначення (1 – Неділя, 7 - Субота) використовують більшість сучасних СКБД та відповідних технологій мов програмування.

2. Створити атрибут weekDay в таблиці WorkerSpecialtySchedule з визначеним типом.

У цьому разі запити суттєво не зміняться, а в них добавиться нова умова відповідності поточного для тижня тому, що виставлено у одиниці розкладу (як це робиться для проміжку часу). *Кінець зауваження.*

*Зауваження.*

Для підвищення швидкодії запитів на вибірку даних, можна виконати індексування ([2; 5]) окремих атрибутів таблиць.

Оскільки окремі запити можуть потребувати різну статистичну інформацію та різну індексацію атрибутів, доцільно клонувати вміст бази та аналізувати запити на різних базах.

*Кінець зауваження.*

*Зауваження*

Якщо, для реалізації деякої функції додатку, необхідно виконати групу запитів як одне ціле (або разом виконуються, або база після збою буде знаходитися в стані до виконання першого оператору), то доцільно використати механізм транзакцій ([3; 4; 6]). Транзакції повинні виконуватися настільки швидко, наскільки це можливо, але з прийнятним рівнем ізоляції.

В курсовій необхідно навести аналіз лише однієї з транзакцій, що використовуються. В аналізі необхідно навести приклади порушення цілісності бази, які виникають при використанні рівнів ізоляції транзакцій, що нижче за серіалізацію. На підставі даних прикладів, дати коротке текстове обґрунтування вибору того чи іншого рівню ізоляції для використання в додатку курсової роботи.

*Кінець зауваження*

*Кінець прикладу.*

### 2.2.3.8.Опис інтерфейсу з користувачем

При розробці інтерфейсу користувача необхідно виконувати керівні принципи UI системи, в межах якої реалізовано програмний додаток.

Користувацький інтерфейс програмного додатку повинен бути дружнім до користувача: принципи використання його елементів повинні бути інтуітивно зрозумілими, вимоги до дій користувача не повинні перевищувати психофізичні обмеження людини, дизайн UI повинен бути приємним.

При описі інтерфейсу користувача необхідно задокументувати властивості всіх його елементів, супроводжуючи опис їх графічним зображенням.

Також необхідно навести схему взаємозв'язків між контейнерними елементами графічного інтерфейсу (вікнами, сторінками, представленнями і т.п.)

Основні правила, принципи та рекомендації по створенню графічного інтерфейсу можна знайти в роботі [5].

Для реалізації інтерфейсу користувача можна використовувати будь-яку технологію мов програмування, однак робити додаток консольним заборонено.

В звіті, для опису інтерфейсу користувача, можна використовувати як його скріншоти так і діаграми UI, виконі в case-засобах.

**Приклад 2.2.3.8.1.**

UI додатку баз даних складу товарів має структуру, наведену в Рис.2.2.3.8.1.

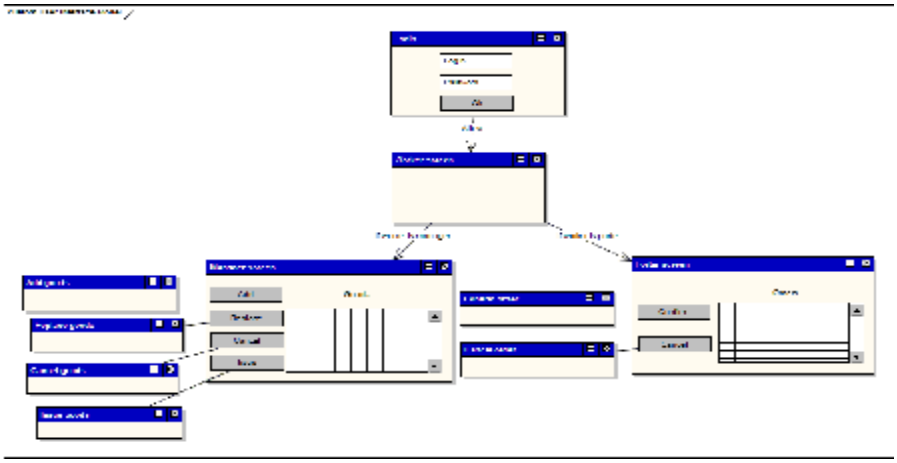

**Рис. 2.2.3.8.1. Структура UI додатку баз даних. Схема виконана в case-засобі Enterprise Architect**

Перед тим, як зайти на свою робочу сторінку, працівник складу повинен пройти процедуру автентифікації. Це забезпечується сторінкою Login, позитивне проходження якої дозволяє перейти на сторінку Worker screen.

Як вже зазначалося раніше, працівник складу може працювати на декількох посадах.

Для переходу на робочу сторінку однієї з своїх посад, він активує її з Worker Screen. На даний момент розроблено Worker screen та Porter screen. Якщо ж працівник не працює на певній посаді, то на відповідну сторінку він перейти не може.

Основними елементами сторінки Manager screen є список товарів на складі з їх характеристиками та можливість активувати сторінки додавання, переміщення, списання та видачі товару – відповідно: Add goods, Replace goods, Cancel goods, Issue goods.

Основними елементами сторінки Porter screen є список наказів йому та можливість активації сторінок підтвердження виконання наказу та його відміна – відповідно: Confirm order, Cancel order

*Зауваження.* Далі в курсовій роботі необхідно привести скріншоти вказаних сторінок та детально їх описати. *Кінець зауваження.*

*Кінець прикладу*

Слід зазначити, що у звіті немає необхідності докладно описувати всі можливі повідомлення сторінок. Достатньо указати їх назву, тип (інформаційне, попереджувальне, помилка) та коротко дати їм харакретистику.

### 2.2.3.9.Приклади використання додатку баз даних

В цьому розділі необхідно навести приклади використання додатку баз даних, в яких розглянуті основні випадки роботи з ним. Приклади необхідно супроводжувати скріншотами UI та детальним їх текстовим розбором.

Додаток баз даних використовується в межах виділених раніше процесів. Тому множину прикладів доцільно обмежити розглядом по одному випадку проходження цих процесів.

В якості прикладу цього можна навести послідовність задіяння елементів UI додатку при проходженні процесу реєстрації та розміщення товару на складі.

### 2.2.4. Висновки по роботі

Висновки по курсовій роботі повинні бути чіткими та лаконічними. У висновках по роботі відображаються основні її результати та значення цих результатів.

З прикладом висновків по роботі можуть слугувати висновки до даних методичних вказівок.

### 2.2.5. Критерії оцінювання роботи

Третім з кінця листом у звіті з курсової роботи є лист «Критерії оцінювання роботи», зміст якого є у Додатку Б.

На даному листі студент відповідним написом від руки підтверджує факт свого ознайомлення та погодження з критеріями оцінювання курсової роботи, ставить поточну дату та власний підпис.

### 2.2.6. Процес захисту роботи

Останні два листа у звіті з курсової роботи є «Процес захисту роботи», зміст якого можна знайти в Додатку В до даних методичних вказівок.

При першій здачі звіту з курсової роботи викладач у відповідному полі аркушу вносить поточну дату та ставить свій підпис. У відповідному полі свій підпис ставить і студент, підтверджуючи дату подачі звіту на перевірку.

Після цього викладач ознайомлюється із змістом звіту по курсовій роботі та відповідним розробленим додатком баз даних. При виникненні по ним зауважень, викладач у відповідні поля аркушу вносить дату та зміст зауваження. Студент ознайомлюється з зауваженнями в присутності викладача та підписує кожне, підтверджуючи свою згоду з ними. Зауваження виправляються студентом і, після

ознайомлення з виправленнями, викладач у відповідних полях ставить поточну дату та підпис, підтверджуючи свою згоду з результатами виправлень зауважень.

Якщо у викладача зауваження по результатам роботи відсутні, або сплив час, відведений під виконання та захист роботи, у відповідні поля він вносить оцінку за роботу, ставить поточну дату та свій підпис.

# ВИСНОВКИ

В даних методичний вказівках виконано визначення основних процесів, які відбуваються під час виконання та захисту курсової роботи та дано докладні роз’яснення по ним. А саме:

1. Виділено елементи структури процесів виконання та захисту курсової роботи та визначено місця, права і обов’язки викладача та студентів в них.

2. Формалізовано юридичну сторону процесів захисту та оцінювання курсової роботи.

3. Визначено зміст звіту по курсовій роботі та дано детальні роз’яснення по його оформленню, які супроводжувалися наочними прикладами.

Чітке слідування студентами та викладачем вказівкам, які надані в даних методичних рекомендаціях повинно спростити процес їх сумісної праці по курсовій роботі та сформувати у студента знання та вміння розробки додатків баз даних.

## ЛІТЕРАТУРА

1. Буч Г., Максимчук Р.А., Энгл М.У., Янг Б.Дж., Коналлен Д., Хьюстон К.А. Объектно-ориентированный анализ и проектирование с примерами приложений, 3-е изд. : Пер. с англ. – М.: ООО «И.Д. Вильямс», 2008. – 720 с.

2. Буч Г., Рамбо Д., Якобсон И. Введение в UML от создателей языка. 2-е изд.: Пер. с англ. Мухин Н. – М.: ДМК Пресс, 2011. – 496 с.

3. Дейт К. Дж. Введение в системы баз данных, 7-е издание.: Пер. с англ. – М.: Издательский дом «Вильямс», 2001. – 1072 с.

4. Крёнке Д. Теория и практика построения баз данных. – СПб.: Питер, 2003. – 800 с.

5. Раскин Д., Интерфейс: новые направления в проектировании компьютерных систем. — Пер. с англ. — СПб: Символ-Плюс, 2004. — 272 с.

6. Ролланд Ф.Д. Основные концепции баз данных. : Пер. с англ. – М.: Издательский дом «Вильямс», 2002. – 256 с.

**ДОДАТОК А**

**ТИТУЛЬНИЙ ЛИСТ ЗВІТУ З КУРСОВОЇ РОБОТИ**

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»
НАВЧАЛЬНО-НАУКОВИЙ КОМПЛЕКС «ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ»

Кафедра математичних методів системного аналізу

# Курсова робота

**з дисципліни: «Організація баз даних та знань»**
**на тему: «»**

**Виконав:**
студент групи КА-
Прізвище Ім’я По батькові

м. Київ – 20__ р.

## ДОДАТОК Б

### КРИТЕРІЇ ОЦІНЮВАННЯ РОБОТИ

В наведеній нижче таблиці, правило, сформульоване в кожному наступному по порядку слідування рядку, обмежує попередні.

| Необхідна умова | Бал за 100-бальною системою | Еквівалент у літерах | Бал за національною шкалою |
|---|---|---|---|
| Студент приступив до процесу захисту роботи, має робочий програмний додаток, але не виправив жодного зауваження викладача | >29 | FX | Незадовільно |
| По спроектованій схемі бази у викладача зауваження відсутні | >59 | E | Задовільно |
| По спроектованим функціональним, структурним схемам, діаграмі класів та структурі звіту з курсової роботи у викладача зауваження відсутні | >64 | D | |
| По організації взаємодії програмного додатку із СКБД у викладача зауваження відсутні | >74 | C | Добре |
| По організації процесу маніпулювання даними у програмному додатку у викладача зауваження відсутні | >84 | B | |
| Ні по програмному додатку, ні по оформленню звіту з курсової у викладача зауважень немає | >94 | A | Відмінно |

(Студентом рукою вноситься запис «З критеріями оцінювання ознайомлений», ставиться поточна дата та підпис)

## ДОДАТОК В

## ПРОЦЕС ЗАХИСТУ РОБОТИ

| (Дата здачі звіту на перевірку) | (Підпис викладача) | (Підпис студента) |
|---|---|---|

### Зауваження по звіту з курсової роботи

| Дата висунення зауваження | Зміст | Підпис студента, як факт підтвердження того, що з зауваженням він ознайомлений та погоджений | Дата виправлення зауваження | Підпис викладача, як факт підтвердження виправлення студентом зауваження |
|---|---|---|---|---|
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

## Зауваження по програмному додатку

| Дата висунення зауваження | Зміст | Підпис студента, як факт підтвердження того, що з зауваженням він ознайомлений та погоджений | Дата виправлення зауваження | Підпис викладача, як факт підтвердження виправлення студентом зауваження |
|---|---|---|---|---|
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

Роботу оцінено на _____ _____ _____

(Оцінка за курсову роботу) (поточна дата) (підпис викладача)

## ДОДАТОК Д

### СПИСОК ТЕМ КУРСОВИХ РОБІТ

Номер теми для виконання курсової роботи можна визначити як остача від цілочисельного ділення на 20 номеру студента за списком групи плюс один.

Якщо студент бажає обрати іншу тему з списку, то він повинен узгодити це з викладачем.

Список тем:

1. Додаток бази даних морського порту.
2. Додаток бази даних бухгалтерії.
3. Додаток бази даних магазину.
4. Додаток бази даних відділу кадрів.
5. Додаток бази даних методичного відділу.
6. Додаток бази даних служби таксі.
7. Додаток бази даних диспетчерської аеропорту.
8. Додаток бази даних центру керування польотами космічних апаратів.
9. Додаток бази даних міліції (поліції).
10. Додаток бази даних міського управління.
11. Додаток бази даних архітектурного відділу.
12. Додаток бази даних туристичної агенції.
13. Додаток бази даних парламенту.
14. Додаток бази даних аналітичної служби.
15. Додаток бази даних редакції журналу.
16. Додаток бази даних ЖЕКу.
17. Додаток бази даних менеджера проектів.
18. Додаток бази даних ресторану.
19. Додаток бази даних логістичної контори.
20. Додаток бази даних HR-відділу.