

## ПРАКТИЧНЕ ЗАНЯТТЯ № 05

### ТЕХНОЛОГІЯ СТВОРЕННЯ ПРОГРАМНИХ ПРОДУКТІВ

**Тема:** ОБ'ЄКТНО-ОРІЄНТОВАНИЙ АНАЛІЗ ТА ОБ'ЄКТНО-ОРІЄНТОВАНЕ ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ: ДІАГРАМА КОМПОНЕНТІВ, ДІАГРАМА РОЗГОРТАННЯ

**Мета:** дослідження предметної області, вивчення процесу побудови діаграми компонентів та розгортання за допомогою інструмента об'єктного моделювання StarUML.

**Програмне забезпечення:** ОС Windows/Linux, StarUML (<https://staruml.io/>).

#### Теоретичний базис:

##### 1. Діаграма компонентів

Діаграма компонентів, також відома як діаграма компонентів UML, описує організацію та з'єднання фізичних компонентів у системі. Діаграми компонентів часто малюють, щоб допомогти змодельовати деталі реалізації та перевірити, чи кожен аспект необхідних функцій системи охоплений запланованою розробкою.

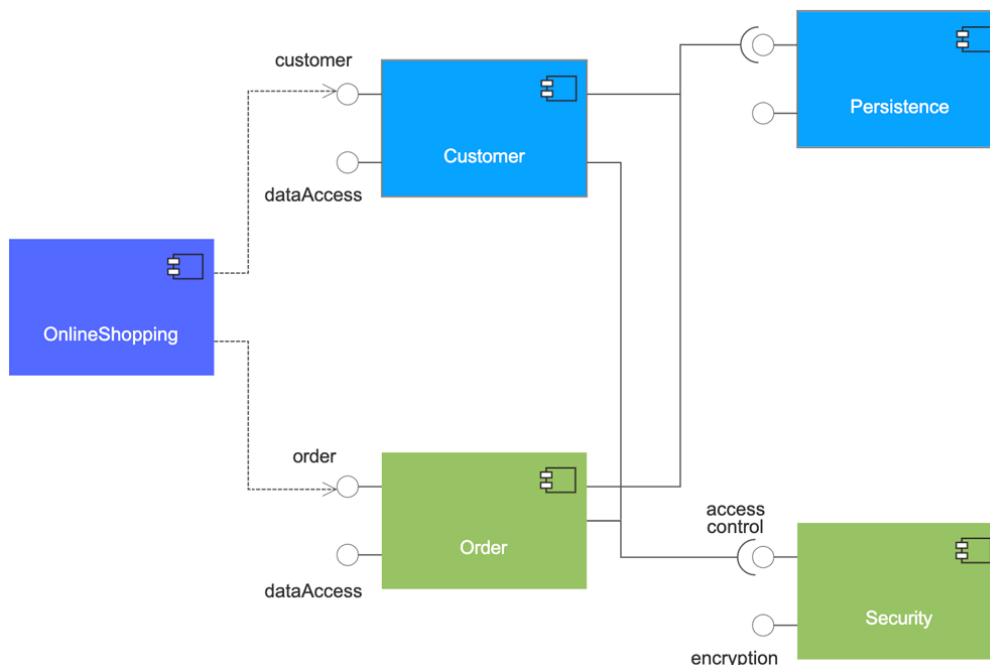

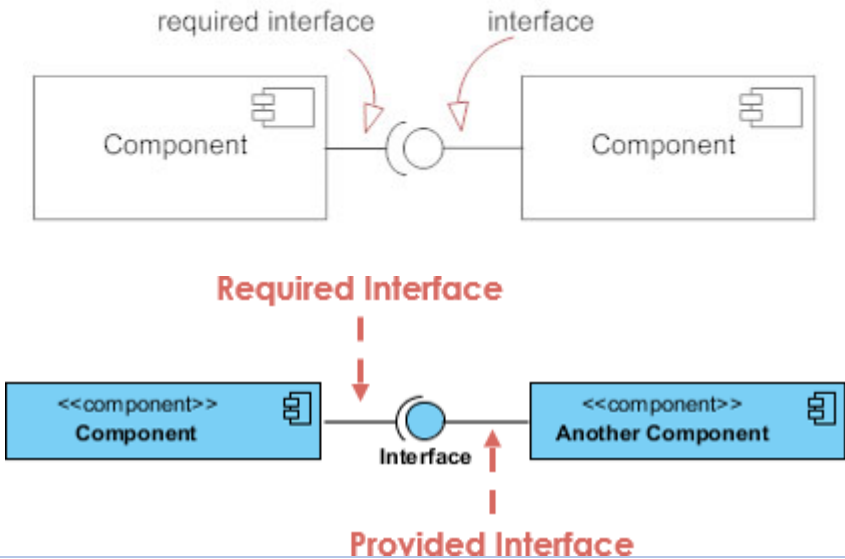
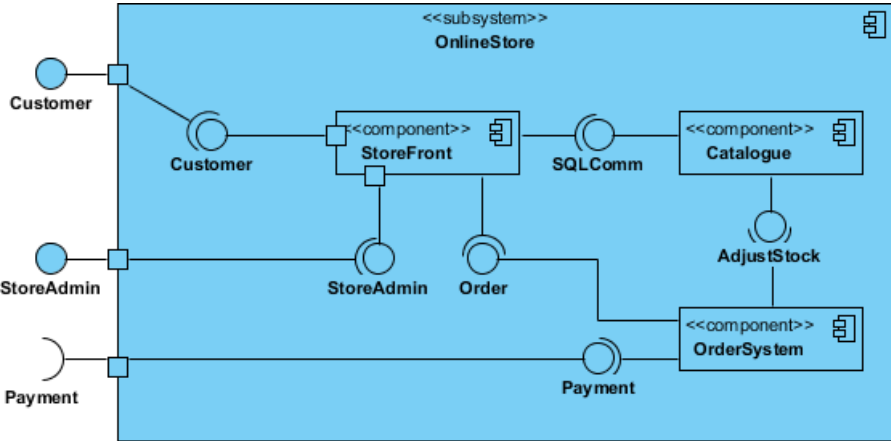
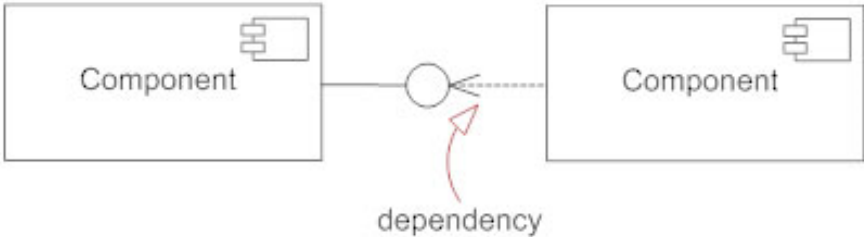


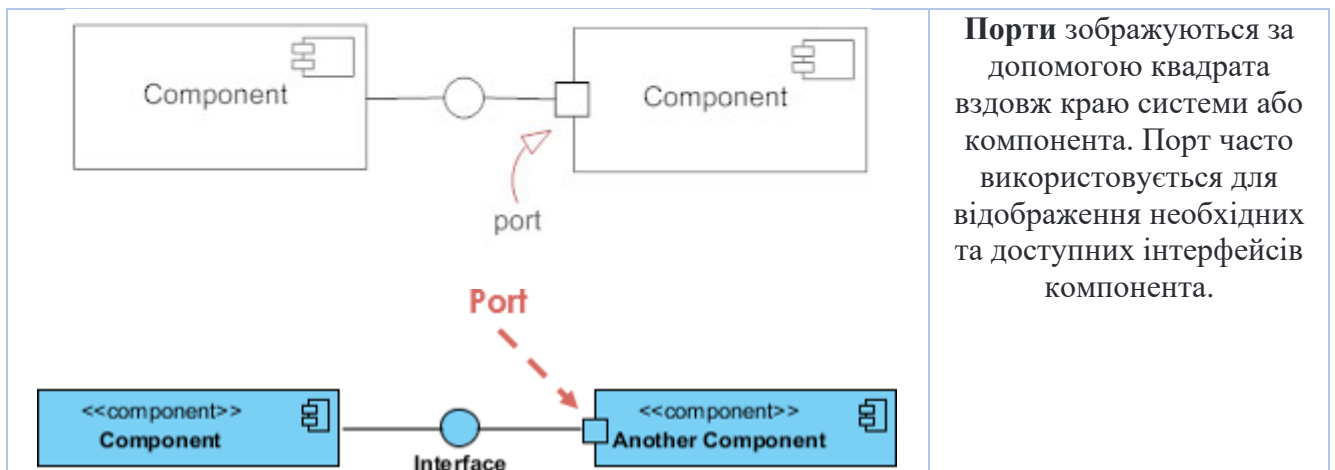
Рис. Приклад діаграми компонентів

#### Основні символи та позначення компонентних схем

Графічно діаграма компонентів - це набір вершин і дуг, які зазвичай містять компоненти, інтерфейси, а також зв'язки залежності, агрегації, обмеження,

узагальнення, асоціації та реалізаційні зв'язки. Вона також може містити примітки та обмеження.

	<p>Компонент - це логічна одиниця системи, трохи вища абстракція, ніж класи. Він зображується у вигляді прямокутника з меншим прямокутником у правому верхньому куті з табуляцією або словом, написаним над назвою компонента, щоб допомогти відрізнити його від класу.</p>
	<p><b>Інтерфейс</b> (маленьке коло або півколо на паличці) описує групу операцій, які використовуються (необхідні) або створюються (надаються) компонентами. Повне коло представляє інтерфейс, створений або наданий компонентом. Напівколо представляє необхідний інтерфейс, наприклад, введення даних людиною.</p>
	<p><b>Класифікатор</b> підсистеми є спеціалізованою версією класифікатора компонента. Через це елемент позначення підсистеми успадковує всі ті самі правила, що й елемент позначення компонента. Єдина відмінність полягає у тому, що елемент нотації підсистеми має ключове слово підсистеми замість компонента.</p>
	<p><b>Залежності</b> між компонентами зображаються за допомогою пунктирних стрілок.</p>



## Приклади діаграми компонентів

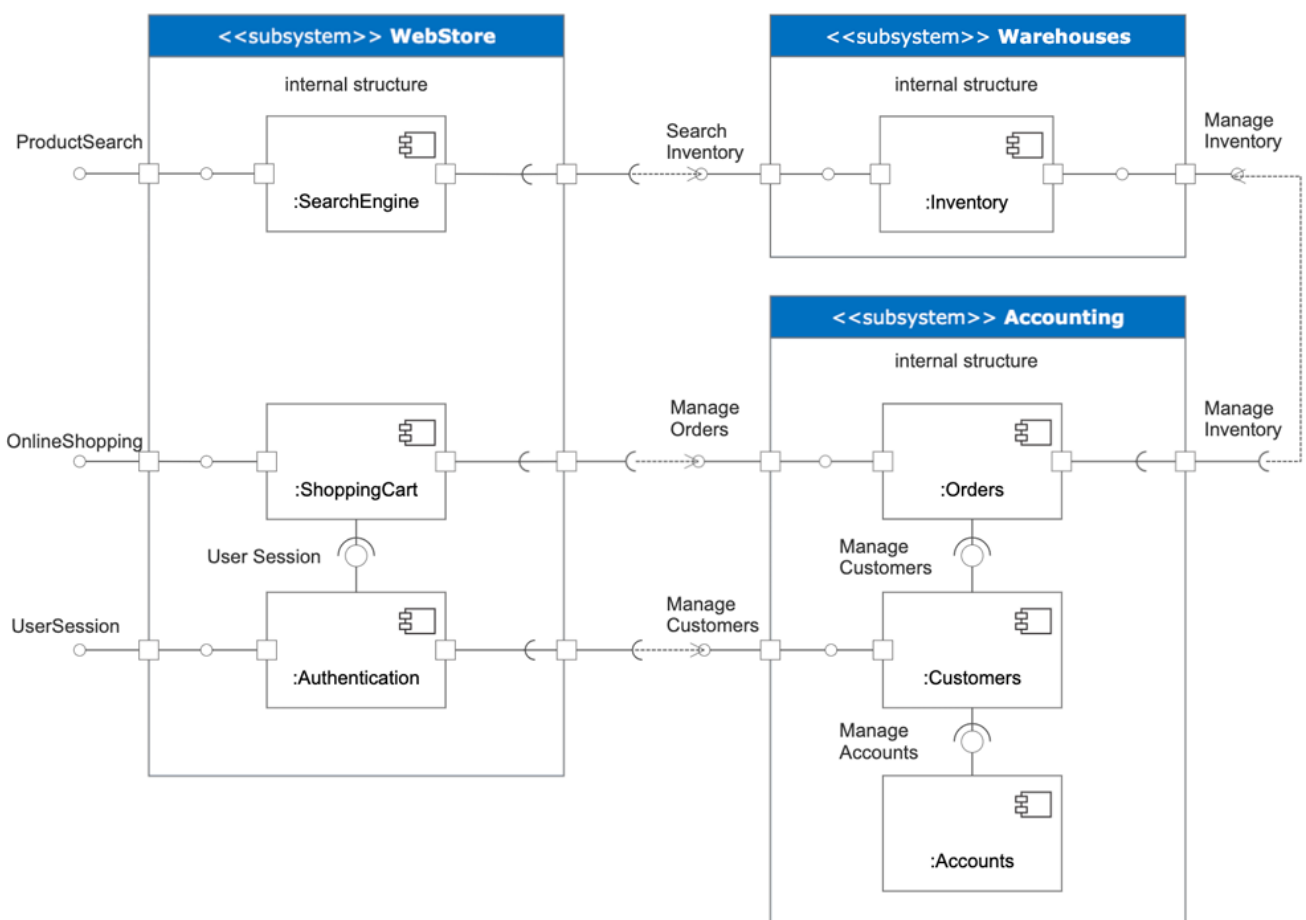


Рис. Приклад діаграми компонентів

## 2. Діаграма розгортання

Діаграма розгортання - це тип діаграми, яка визначає фізичне обладнання, на якому буде працювати програмна система. Вона також визначає, як програмне забезпечення розгортається на базовому обладнанні. Вона відображає програмні частини системи на пристрої, які будуть їх виконувати.

Діаграма розгортання відображає архітектуру програмного забезпечення, створену під час проектування, на фізичну архітектуру системи, яка його виконує.

У розподілених системах вона моделює розподіл програмного забезпечення між фізичними вузлами.

Програмні системи відображаються за допомогою різних артефактів, а потім вони відображаються на середовище виконання, яке буде виконувати програмне забезпечення, наприклад, на вузли.

Існує дві форми діаграми розгортання:

- Дескрипторна форма або форма на рівні специфікацій (містить вузли, зв'язки між вузлами та артефакти)
- Форма екземплярів (містить екземпляри вузлів, зв'язки між вузлами та екземпляри артефактів)

### Діаграма розгортання. Символи та позначення

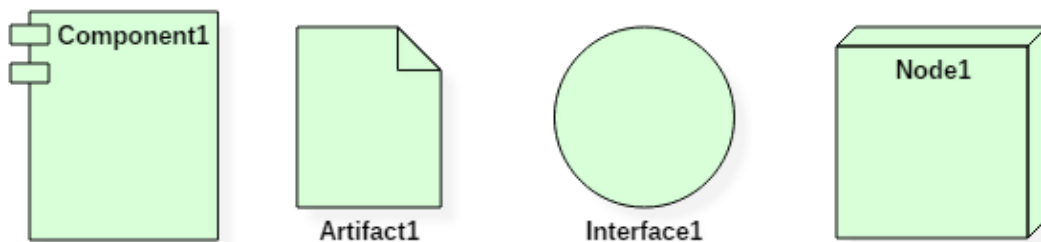


Рис. Deployment Diagram Notations

Діаграма розгортання складається з наступних позначень:

- Вузол
- Компонент
- Артефакт
- Інтерфейс

**Артефакт** являє собою специфікацію конкретної реальної сутності, пов'язаної з розробкою програмного забезпечення. Ви можете використовувати артефакт для опису фреймворку, який використовується в процесі розробки програмного забезпечення, або виконуваного файлу. Артефакти розгортаються на вузлах. Найпоширенішими артефактами є наступні:

- Вихідні файли
- Виконавчі файли
- Таблиці бази даних
- Скрипти
- Файли DLL
- Посібники користувача або документація
- Вихідні файли



Рис. Графічна нотація артефакту

Екземпляр артефакту - це примірник певного артефакту. Екземпляр артефакту позначається тим самим символом, що й артефакт, за винятком того, що ім'я підкреслюється. Діаграма UML дозволяє відрізнити оригінальний артефакт від екземпляра. Кожна фізична копія або файл є екземпляром унікального артефакту.



Рис. Графічна нотація артефакту

**Нода** - це обчислювальний ресурс, на якому розгортаються артефакти для виконання. Нода - це фізичний об'єкт, який може виконувати один або декілька артефактів. Розмір ноди може змінюватися в залежності від розміру проекту.

Як правило, вузол має два стереотипи, які наведені нижче:

<< **device** >> Це вузол, який представляє собою фізичну машину, здатну виконувати обчислення. Пристроєм може бути роутер або серверний комп'ютер. Він представляється за допомогою вузла зі стереотипом <<device>>.

У моделі UML можете вкладати один або декілька пристроїв один в одного.



Рис. Графічна нотація ноди

<< **execution environment** >> Це вузол, який представляє середовище, в якому буде виконуватися програмне забезпечення. Наприклад, програми на Java виконуються у віртуальній машині Java (JVM). JVM розглядається як середовище виконання для Java-додатків.

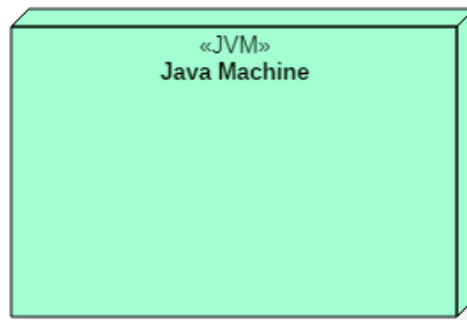


Рис. Execution environment node

## Приклади діаграм розгортання

### Прояв компонентів через артефакти

У той час як діаграми компонентів показують компоненти і зв'язки між компонентами і класифікаторами, а діаграми розгортання - розгортання артефактів до цілей розгортання, відсутня проміжна діаграма - діаграма маніфестації, яка використовується для відображення прояву (реалізації) компонентів артефактами і внутрішньої структури артефактів.

Оскільки діаграми маніфестації не визначені специфікацією UML 2.4, маніфестація компонентів артефактами може бути показана за допомогою або діаграм компонентів, або діаграм розгортання.

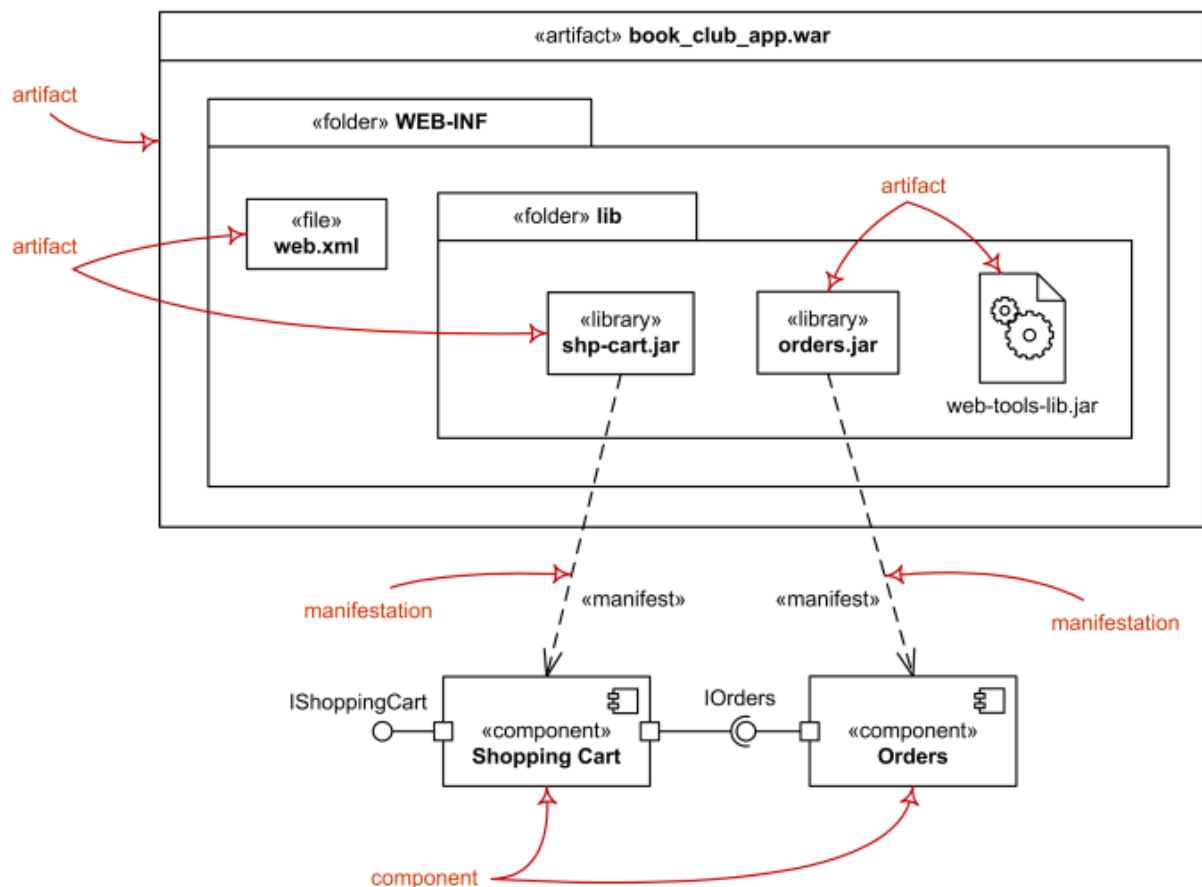


Рис. Manifestation of components by artifacts.

### Діаграма розгортання на рівні специфікацій

Діаграма розгортання на рівні специфікацій (також звана діаграмою на рівні типів) показує деякий огляд розгортання артефактів до цілей розгортання, без посилання на конкретні екземпляри артефактів або вузлів.

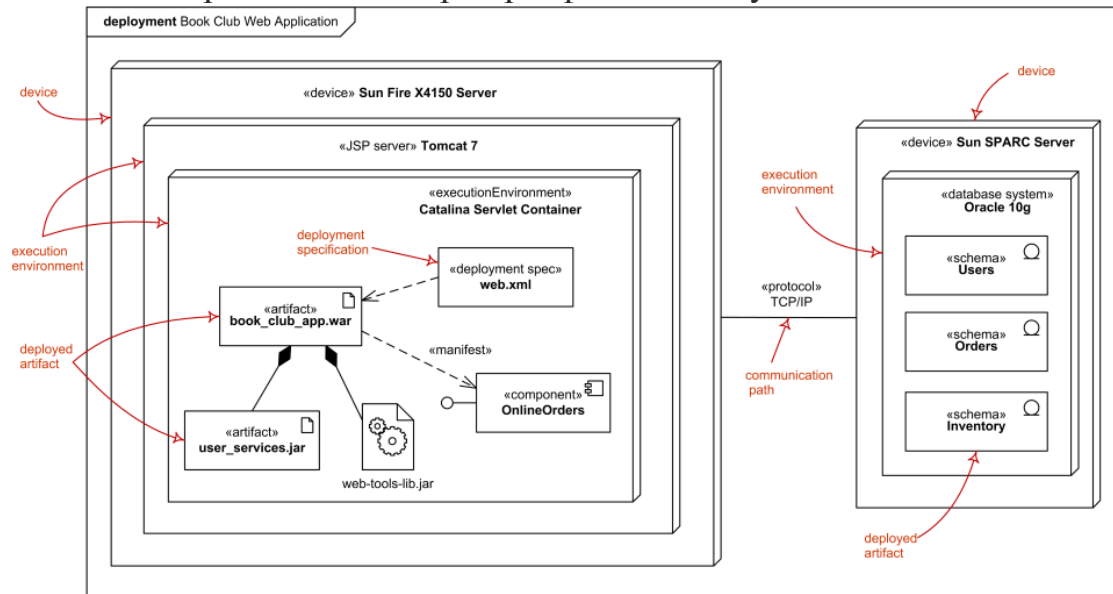


Рис. Specification level deployment diagram - web application deployed to Tomcat JSP server and database schemas - to database system.

### Схема розгортання на рівні екземплярів

Діаграма розгортання на рівні екземплярів показує розгортання екземплярів артефактів до конкретних екземплярів цілей розгортання. Її можна використовувати, наприклад, щоб показати відмінності у розгортанні до середовищ розробки, тестування або виробництва із зазначенням імен/ідентифікаторів конкретних серверів або пристроїв розгортання.

У наведеному нижче прикладі веб-додаток розгорнуто на сервері додатків wsrv-01, а кілька схем баз даних - на сервері баз даних dbsrv-14.

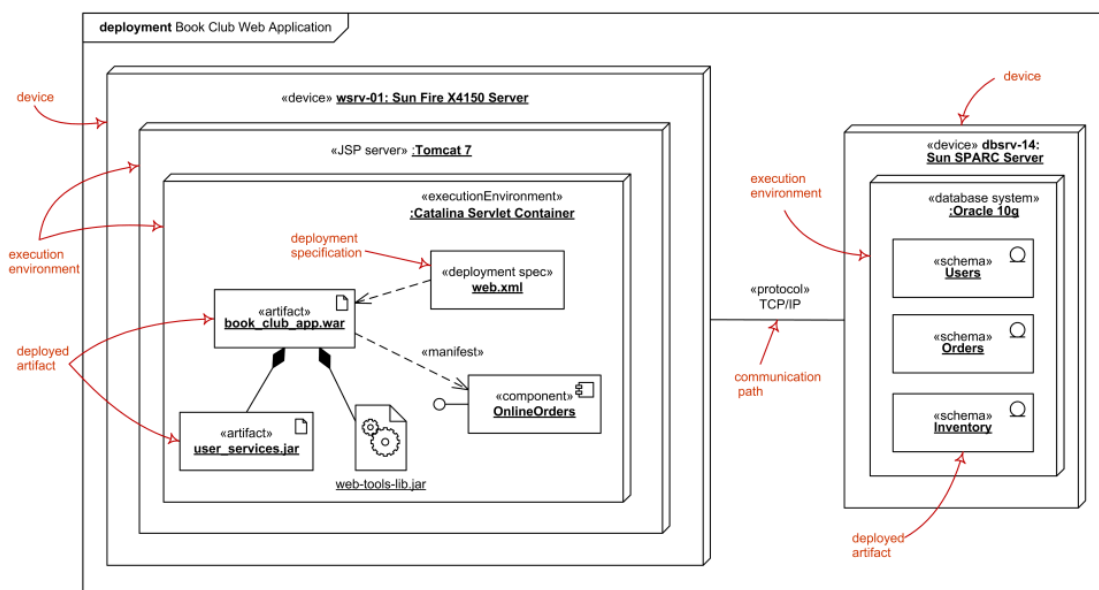


Рис. Instance level deployment diagram - web application deployed to Tomcat JSP server and database schemas - to database system.

### **Порядок виконання роботи:**

1. У відповідності до власного варіанту індивідуального завдання здійснити побудову діаграми компонентів у середовищі StarUML.
2. У відповідності до власного варіанту індивідуального завдання здійснити побудову пакет діаграм розгортання у середовищі StarUML.

### **Контрольні питання:**

1. Які види елементів моделі представлені на діаграмі компонентів?
2. Як пов'язані між собою діаграми пакетів і діаграми компонентів?
3. Що показує діаграма розміщення?
4. Які суті відображуються на діаграмах-розміщення?
5. У яких випадках необхідне застосування діаграм розміщення?
6. Чому потрібно будувати різні діаграми при моделюванні системи?
7. Які діаграми відповідають статичному поданням про систему?
8. Основні нотації на діаграмах компонентів та діаграмах розгортання?

### **Література:**

1. Grady Booch, James Rumbaugh, Ivar Jacobson. Unified Modeling Language User Guide. Addison-Wesley, 1999. 496 p. ISBN: 0-201-57168-4.
2. Martin Fowler. UML Distilled: A Brief Guide to the Standard Object Modeling Language. Addison-Wesley, 2004. 208 p. ISBN: 0-321-19368-7.
3. Craig Larman. Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development. Prentice Hall, 2004. 736 p. ISBN: 0-13-148906-2.
4. Grady Booch, James Rumbaugh, Ivar Jacobson. The Unified Modeling Language Reference Manual. Addison-Wesley, 1999. 720 p. ISBN: 0-201-30998-X.
5. Doug Rosenberg, Kendall Scott. Applying Use Case Driven Object Modeling with UML: An Annotated e-Commerce Example. Addison-Wesley, 2001. 512 p. ISBN: 0-201-60489-7.
6. Jean-Marc Nerson. Object-Oriented Software Construction. Prentice Hall, 2002. 704 p. ISBN: 0-13-629155-4.
7. Jim Arlow, Ila Neustadt. UML 2 and the Unified Process: Practical Object-Oriented Analysis and Design. Addison-Wesley, 2005. 624 p. ISBN: 0-321-26779-8.
8. Michael Blaha, James Rumbaugh. Object-Oriented Modeling and Design with UML. Prentice Hall, 2004. 416 p. ISBN: 0-13-196845-0.
9. Alan Dennis, Barbara Haley Wixom, David Tegarden. Systems Analysis and Design: An Object-Oriented Approach with UML. Wiley, 2014. 544 p. ISBN: 978-1-118-56497-6.
10. Craig Larman. UML 2 and the Unified Process: Practical Object-Oriented Analysis and Design. Prentice Hall, 2005. 736 p. ISBN: 0-13-148906-2.
11. Richard F. Schmidt. Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development. Pearson, 2019. 480 p. ISBN: 978-0131489066.
12. Dan Pilone, Neil Pitman. UML 2.0 in a Nutshell. O'Reilly Media, 2005. 258 p. ISBN: 0-596-00795-7.
13. Doug Rosenberg, Matt Stephens. Use Case Driven Object Modeling with UML: A Practical Approach. Apress, 2007. 648 p. ISBN: 978-1-59059-774-3.