Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ОДЕСЬКА ПОЛІТЕХНІКА”
Інститут комп'ютерних систем
Кафедра інформаційних систем

ПРОЕКТУВАННЯ ІНФОРМАЦІЙНИХ СИСТЕМ

Методичні вказівки до лабораторних робіт

Затверджено на засіданні кафедри ІС
Протокол №1 , від 1.09.2023 р.
Зав.каф. інформаційних систем

_____

Проф.д.т.н. Арсірій О.О.

Одеса: НУОП, 2023

Методичні вказівки до лабораторних робіт з дисципліни " Проектування інформаційних систем " розроблений для студентів інституту комп'ютерних систем усіх форм навчання за спеціальністю 122 - Комп'ютерні науки / Укладачі: О. М. Галчонков, І.О.Баськов, А.В.Денисенко, М.І.Бабіч. - Одеса: НУОП, 2023. - 26с.
Освітня програма - Комп'ютерні науки. Освітньо-професійна програма: 2022.
Укладачі: Галчонков О.М., канд. техн. наук, доц., Баськов І.О., старший викладач, Денисенко А.В., канд. техн. наук, старший викладач, Бабіч М.І., канд. техн. наук, старший викладач.

О. М. Галчонков, І.О.Баськов, А.В.Денисенко, М.І.Бабіч. 122 – Комп'ютерні науки. **Методичні вказівки до лабораторних робіт з дисципліни "Проектування інформаційних систем ".** Методичні вказівки до лабораторних робіт призначені для закріплення навичок проектування програмного забезпечення шляхом побудови UML моделі у вигляді сукупності діаграм..

Лабораторні роботи

Із самого початку семестру необхідно вибрати тему кваліфікаційної роботи та керівника, домовитися з керівником та оформити це відповідною заявою, яку треба завізувати у керівника.

Приклади тем можна взяти із наказу на затвердження тем цього чи минулого року.

Мета: максимально нагадати студентам як робляться діаграми для пояснювальної записки та максимально їх просунути в написанні власної пояснювальної записки.

Лабораторні роботи 1,…,8 робляться з урахуванням прикладів пояснювальних записок кваліфікаційних робіт, наведених у додатках 1,2,3,4. Для кожної лабораторної роботи вибирається свій додаток та своя діаграма.

Обов'язковою умовою є розробка додаткових функціональних вимог, що спричинить переробку діаграм, наведених у додатках.

Протокол лабораторної роботи повинен містити:

1. Титульний

2. Функціонал додатка, вибраного за основу

3. Доданий функціонал.

4. Діаграма (з урахуванням повного функціоналу) з поясненнями

5. Висновки

Допускається перехід, починаючи з будь-якого номера лабораторної роботи, на розробку відповідних діаграм із власної кваліфікаційної роботи. У цьому, кількість лабораторних робіт має бути щонайменше 15 за рахунок побудови додаткових діаграм (діаграми послідовностей, діяльності, інтерфейсів тощо.).

Для побудови діаграм треба використовувати Star UML або аналогічні. Прототипування треба робити на Figma чи аналогічних (Це он-лайн сервіс, всі інструкції на сайті).

Лабораторна робота № 1

“ Створення діаграми прецедентів для інформаційної системи підтримки туристичних подорожей ” .

Мета роботи.

Оволодіти навичками зі створення нового проекту і побудові діаграми прецедентів

Теоретичні положення.

Ектор (actor) - це безліч логічно пов'язаних ролей, виконуваних при взаємодії з прецедентами або сутностями (система, підсистема або клас). Ектором може бути людина або інша система, підсистема або клас, які представляють щось поза суті.

Прецедент (use case) - опис безлічі послідовних подій (включаючи варіанти), що виконуються системою, які призводять до спостережуваного Ектором результату. Прецедент являє поведінку суті, описуючи взаємодію між Ектор і системою. Прецедент не показує, "як" досягається певний результат, а тільки "що" саме виконується.

Завдання до роботи.

1. Створення нового проекту в StarUML
2. Постановка завдання. Визначення робочої області моделювання.
3. Діаграма варіантів використання.

Способ обробки результатів.

Необхідно придумати додатковий функціонал до існуючого проекту, пояснити його доцільність щодо наявного функціоналу, та розробити діаграму відповідно до повного функціонала.

Домашне завдання.

Підберіть найбільш підходящий фреймворк для реалізації додаткового введеного функціоналу

Рекомендації щодо оформлення звіту з роботи.

Оформлення у суворій відповідності до [10].

Контрольні запитання за темою роботи.

1. Як зміниться діаграма, якщо використовувати архітектуру мікросервісів.

2.Як зміниться діаграма, якщо використовувати мікрофронтенд в рамках мікросервісної архітектури.

3. Як зміниться діаграма, якщо використовувати вбудоване тестування під час використання програми.

4. Як зміниться діаграма, якщо використовуватиме методи підвищення надійності роботи.

Література [1,…,10]

Лабораторна робота № 2

“ Створення діаграми прецедентів для інформаційної системи онлайн оренди автомобілів”.

Мета роботи.

Оволодіти навичками зі створення нового проекту і побудові діаграми прецедентів

Теоретичні положення.

Ектор (actor) - це безліч логічно пов'язаних ролей, виконуваних при взаємодії з прецедентами або сутностями (система, підсистема або клас). Ектором може бути людина або інша система, підсистема або клас, які представляють щось поза суті.

Прецедент (use case) - опис безлічі послідовних подій (включаючи варіанти), що виконуються системою, які призводять до спостережуваного Ектором результату. Прецедент являє поведінку суті, описуючи взаємодію між Ектор і системою. Прецедент не показує, "як" досягається певний результат, а тільки "що" саме виконується.

Завдання до роботи.

1. Створення нового проекту в StarUML

2. Постановка завдання. Визначення робочої області моделювання.

3. Діаграма варіантів використання.

Способ обробки результатів.

Необхідно придумати додатковий функціонал до існуючого проекту, пояснити його доцільність щодо наявного функціоналу, та розробити діаграму відповідно до повного функціонала.

Домашне завдання.

Підберіть найбільш підходящий фреймворк для реалізації додаткового введеного функціоналу

Рекомендації щодо оформлення звіту з роботи.

Оформлення у суворій відповідності до [10].

Контрольні запитання за темою роботи.

1. Як зміниться діаграма, якщо використовувати архітектуру мікросервісів.
2. Як зміниться діаграма, якщо використовувати мікрофронтенд в рамках мікросервісної архітектури.
3. Як зміниться діаграма, якщо використовувати вбудоване тестування під час використання програми.
4. Як зміниться діаграма, якщо використовуватиме методи підвищення надійності роботи.

Література [1,…,10]

Лабораторна робота № 3

“ Створення діаграми прецедентів для інформаційної системи управління готелем”.

Мета роботи.

Оволодіти навичками зі створення нового проекту і побудові діаграми прецедентів

Теоретичні положення.

Ектор (actor) - це безліч логічно пов'язаних ролей, виконуваних при взаємодії з прецедентами або сутностями (система, підсистема або клас). Ектором може бути людина або інша система, підсистема або клас, які представляють щось поза суті.

Прецедент (use case) - опис безлічі послідовних подій (включаючи варіанти), що виконуються системою, які призводять до спостережуваного Ектором результату. Прецедент являє поведінку суті, описуючи взаємодію між Ектор і системою. Прецедент не показує, "як" досягається певний результат, а тільки "що" саме виконується.

Завдання до роботи.

1. Створення нового проекту в StarUML
2. Постановка завдання. Визначення робочої області моделювання.

3. Діаграма варіантів використання.

Способ обробки результатів.

Необхідно придумати додатковий функціонал до існуючого проекту, пояснити його доцільність щодо наявного функціоналу, та розробити діаграму відповідно до повного функціонала.

Домашне завдання.

Підберіть найбільш підходящий фреймворк для реалізації додаткового введеного функціоналу

Рекомендації щодо оформлення звіту з роботи.

Оформлення у суворій відповідності до [10].

Контрольні запитання за темою роботи.

1. Як зміниться діаграма, якщо використовувати архітектуру мікросервісів.
2. Як зміниться діаграма, якщо використовувати мікрофронтенд в рамках мікросервісної архітектури.
3. Як зміниться діаграма, якщо використовувати вбудоване тестування під час використання програми.
4. Як зміниться діаграма, якщо використовуватиме методи підвищення надійності роботи.

Література [1,…,10]

Лабораторна робота № 4

“ Прототипування інтерфейсів користувача для інформаційної системи підтримки туристичних подорожей ”.

Мета роботи.

Набути навичок розробки інтерфейсів користувача

Теоретичні положення.

Грамотно спроектований інтерфейс користувача дуже важливий для успішної роботи системи. Складний у використанні інтерфейс, як мінімум, призводить до помилок користувача. Іноді вони просто відмовляються працювати з програмною системою, незважаючи на її функціональні можливості. Якщо інформація видається плутано чи непослідовно, користувачі можуть зрозуміти її неправильно, внаслідок

чого їх подальші дії можуть призвести до пошкодження даних або навіть до збою в роботі системи.

Наразі майже всі користувачі працюють на персональних комп'ютерах. Усі сучасні персональні комп'ютери підтримують графічний інтерфейс користувача (graphical user interface – GUI), який

передбачає використання кольорового графічного екрану з високою роздільною здатністю і дозволяє працювати з мишею та з клавіатурою.

Хоча текстові інтерфейси ще досить широко застосовуються, особливо в системах, що успадковуються, в наш час користувачі воліють працювати з графічним інтерфейсом.

Завдання до роботи.

1. Розробка додаткових функціональних вимог.
2. Коригування старих графічних інтерфейсів.
3. Розробка нових додаткових графічних інтерфейсів.

Способ обробки результатів.

Необхідно придумати додатковий функціонал до існуючого проекту, пояснити його доцільність щодо наявного функціоналу, та розробити діаграму відповідно до повного функціонала.

Домашне завдання.

Підберіть найбільш підходящий фреймворк для реалізації додаткового введеного функціоналу

Рекомендації щодо оформлення звіту з роботи.

Оформлення у суворій відповідності до [10].

Контрольні запитання за темою роботи.

1. Як зміниться діаграма, якщо використовувати архітектуру мікросервісів.
2. Як зміниться діаграма, якщо використовувати мікрофронтенд в рамках мікросервісної архітектури.
3. Як зміниться діаграма, якщо використовувати вбудоване тестування під час використання програми.
4. Як зміниться діаграма, якщо використовуватиме методи підвищення надійності роботи.

Література [1,…,10]

Лабораторна робота № 5

“ Прототипування інтерфейсів користувача для інформаційної системи онлайн оренди автомобілів ”.

Мета роботи.

Набути навичок розробки інтерфейсів користувача

Теоретичні положення.

Грамотно спроектований інтерфейс користувача дуже важливий для успішної роботи системи. Складний у використанні інтерфейс, як мінімум, призводить до помилок користувача. Іноді вони просто відмовляються працювати з програмною системою, незважаючи на її функціональні можливості. Якщо інформація видається плутано чи непослідовно, користувачі можуть зрозуміти її неправильно, внаслідок чого їх подальші дії можуть призвести до пошкодження даних або навіть до збою в роботі системи.

Наразі майже всі користувачі працюють на персональних комп'ютерах. Усі сучасні персональні комп'ютери підтримують графічний інтерфейс користувача (graphical user interface – GUI), який

передбачає використання кольорового графічного екрану з високою роздільною здатністю і дозволяє працювати з мишею та з клавіатурою.

Хоча текстові інтерфейси ще досить широко застосовуються, особливо в системах, що успадковуються, в наш час користувачі воліють працювати з графічним інтерфейсом.

Завдання до роботи.

1. Розробка додаткових функціональних вимог.
2. Коригування старих графічних інтерфейсів.
3. Розробка нових додаткових графічних інтерфейсів.

Способ обробки результатів.

Необхідно придумати додатковий функціонал до існуючого проекту, пояснити його доцільність щодо наявного функціоналу, та розробити діаграму відповідно до повного функціонала.

Домашне завдання.

Підберіть найбільш підходящий фреймворк для реалізації додаткового введеного функціоналу.

Рекомендації щодо оформлення звіту з роботи.

Оформлення у суворій відповідності до [10].

Контрольні запитання за темою роботи.

1. Як зміниться діаграма, якщо використовувати архітектуру мікросервісів.

2.Як зміниться діаграма, якщо використовувати мікрофронтенд в рамках мікросервісної архітектури.

3. Як зміниться діаграма, якщо використовувати вбудоване тестування під час використання програми.

4. Як зміниться діаграма, якщо використовуватиме методи підвищення надійності роботи.

Література [1,…,10]

Лабораторна робота № 6

“ Прототипування інтерфейсів користувача для інформаційної системи управління готелем ”.

Мета роботи.

Набути навичок розробки інтерфейсів користувача

Теоретичні положення.

Грамотно спроектований інтерфейс користувача дуже важливий для успішної роботи системи. Складний у використанні інтерфейс, як мінімум, призводить до помилок користувача. Іноді вони просто відмовляються працювати з програмною системою, незважаючи на її функціональні можливості. Якщо інформація видається плутано чи непослідовно, користувачі можуть зрозуміти її неправильно, внаслідок чого їх подальші дії можуть призвести до пошкодження даних або навіть до збою в роботі системи.

Наразі майже всі користувачі працюють на персональних комп'ютерах. Усі сучасні персональні комп'ютери підтримують графічний інтерфейс користувача (graphical user interface – GUI), який

передбачає використання кольорового графічного екрану з високою роздільною здатністю і дозволяє працювати з мишею та з клавіатурою.

Хоча текстові інтерфейси ще досить широко застосовуються, особливо в системах, що успадковуються, в наш час користувачі воліють працювати з графічним інтерфейсом.

Завдання до роботи.

1. Розробка додаткових функціональних вимог.

2. Коригування старих графічних інтерфейсів.
3. Розробка нових додаткових графічних інтерфейсів.

Способ обробки результатів.
Необхідно придумати додатковий функціонал до існуючого проекту, пояснити його доцільність щодо наявного функціоналу, та розробити діаграму відповідно до повного функціонала.

Домашне завдання.
Підберіть найбільш підходящий фреймворк для реалізації додаткового введеного функціоналу.

Рекомендації щодо оформлення звіту з роботи.
Оформлення у суворій відповідності до [10].

Контрольні запитання за темою роботи.
1. Як зміниться діаграма, якщо використовувати архітектуру мікросервісів.
2.Як зміниться діаграма, якщо використовувати мікрофронтенд в рамках мікросервісної архітектури.
3. Як зміниться діаграма, якщо використовувати вбудоване тестування під час використання програми.
4. Як зміниться діаграма, якщо використовуватиме методи підвищення надійності роботи.

Література [1,…,10]

Лабораторна робота № 7

“ Діаграма логічного представлення та розгортання системи підтримки туристичних подорожей ”.

Мета роботи.
Набути навичок розробки діаграми логічного представлення та розгортання

Теоретичні положення.
Корпоративні додатки часто вимагають для своєї роботи деякої IT-інфраструктури, зберігають інформацію в базах даних, розташованих десь на серверах компанії, викликають веб-сервіси, використовують загальні ресурси і т. Д. У таких випадках непогано б мати графічне представлення інфраструктури, на яку буде розгорнуто

додаток. Ось для цього і потрібні діаграми розгортання, які іноді називають діаграмами розміщення.

Завдання до роботи.

1. Розробка додаткових функціональних вимог.
2. Розробка нової схеми логічного уявлення

Способ обробки результатів.

Необхідно придумати додатковий функціонал до існуючого проекту, пояснити його доцільність щодо наявного функціоналу, та розробити діаграму відповідно до повного функціонала.

Домашне завдання.

Підберіть найбільш підходящий фреймворк для реалізації додаткового введеного функціоналу

Рекомендації щодо оформлення звіту з роботи.

Оформлення у суворій відповідності до [10].

Контрольні запитання за темою роботи.

1. Як зміниться діаграма, якщо використовувати архітектуру мікросервісів.
2. Як зміниться діаграма, якщо використовувати мікрофронтенд в рамках мікросервісної архітектури.
3. Як зміниться діаграма, якщо використовувати вбудоване тестування під час використання програми.
4. Як зміниться діаграма, якщо використовуватиме методи підвищення надійності роботи.

Література [1,…,10]

Лабораторна робота № 8

“ Діаграма логічного представлення та розгортання системи оренди автомобілів ”.

Мета роботи.

Набути навичок розробки діаграми логічного представлення та розгортання

Теоретичні положення.

Корпоративні додатки часто вимагають для своєї роботи деякої IT-інфраструктури, зберігають інформацію в базах даних, розташованих десь на серверах компанії, викликають веб-сервіси, використовують загальні ресурси і т. Д. У таких випадках непогано б мати графічне представлення інфраструктури, на яку буде розгорнуто додаток. Ось для цього і потрібні діаграми розгортання, які іноді називають діаграмами розміщення.

Завдання до роботи.

1. Розробка додаткових функціональних вимог.
2. Розробка нової схеми логічного уявлення

Способ обробки результатів.

Необхідно придумати додатковий функціонал до існуючого проекту, пояснити його доцільність щодо наявного функціоналу, та розробити діаграму відповідно до повного функціонала.

Домашне завдання.

Підберіть найбільш підходящий фреймворк для реалізації додаткового введеного функціоналу

Рекомендації щодо оформлення звіту з роботи.

Оформлення у суворій відповідності до [10].

Контрольні запитання за темою роботи.

1. Як зміниться діаграма, якщо використовувати архітектуру мікросервісів.
2. Як зміниться діаграма, якщо використовувати мікрофронтенд в рамках мікросервісної архітектури.
3. Як зміниться діаграма, якщо використовувати вбудоване тестування під час використання програми.
4. Як зміниться діаграма, якщо використовуватиме методи підвищення надійності роботи.

Література [1,…,10]

Лабораторна робота № 9

“ Діаграма логічного представлення та розгортання системи управління готелем ”.

Мета роботи.

Набути навичок розробки діаграми логічного представлення та розгортання

Теоретичні положення.

Корпоративні додатки часто вимагають для своєї роботи деякої IT-інфраструктури, зберігають інформацію в базах даних, розташованих десь на серверах компанії, викликають веб-сервіси, використовують загальні ресурси і т. Д. У таких випадках непогано б мати графічне представлення інфраструктури, на яку буде розгорнуто додаток. Ось для цього і потрібні діаграми розгортання, які іноді називають діаграмами розміщення.

Завдання до роботи.

1. Розробка додаткових функціональних вимог.
2. Розробка нової схеми логічного уявлення

Способ обробки результатів.

Необхідно придумати додатковий функціонал до існуючого проекту, пояснити його доцільність щодо наявного функціоналу, та розробити діаграму відповідно до повного функціонала.

Домашне завдання.

Підберіть найбільш підходящий фреймворк для реалізації додаткового введеного функціоналу

Рекомендації щодо оформлення звіту з роботи.

Оформлення у суворій відповідності до [10].

Контрольні запитання за темою роботи.

1. Як зміниться діаграма, якщо використовувати архітектуру мікросервісів.
2. Як зміниться діаграма, якщо використовувати мікрофронтенд в рамках мікросервісної архітектури.
3. Як зміниться діаграма, якщо використовувати вбудоване тестування під час використання програми.
4. Як зміниться діаграма, якщо використовуватиме методи підвищення надійності роботи.

Література [1,…,10]

Лабораторна робота № 10

“ Проектування відносин між класами системи оренди автомобілів ”.

Мета роботи.

Набути навичок розробки діаграми класів

Теоретичні положення.

Клас (class) - категорія речей, які мають загальні атрибути та операції.

Класи - це будівельні блоки будь-об'єктно-орієнтованої системи. Вони являють собою опис сукупності об'єктів із загальними атрибутами, операціями, відносинами і семантикою. При проектуванні об'єктно-орієнтованих систем діаграми класів обов'язкові.

Діаграма класів - це набір статичних, декларативних елементів моделі. Діаграма класів можуть застосовуватися і при прямому проектуванні, тобто в процесі розробки нової системи, і при зворотному проектуванні - описі існуючих і використовуваних систем. Інформація з діаграми класів безпосередньо відображається в вихідний код програми - в більшості існуючих інструментів UML-моделювання можлива кодогенерацію для певної мови програмування (зазвичай Java або C ++). Таким чином, діаграма класів - кінцевий результат проектування і відправна точка процесу розробки.

Завдання до роботи.

1. Розробка додаткових функціональних вимог.
2. Розробка нової діаграми класів.

Способ обробки результатів.

Необхідно придумати додатковий функціонал до існуючого проекту, пояснити його доцільність щодо наявного функціоналу, та розробити діаграму відповідно до повного функціонала.

Домашне завдання.

Підберіть найбільш підходящий фреймворк для реалізації додаткового введеного функціоналу

Рекомендації щодо оформлення звіту з роботи.

Оформлення у суворій відповідності до [10].

Контрольні запитання за темою роботи.

1. Як зміниться діаграма, якщо використовувати архітектуру мікросервісів.

2.Як зміниться діаграма, якщо використовувати мікрофронтенд в рамках мікросервісної архітектури.

3. Як зміниться діаграма, якщо використовувати вбудоване тестування під час використання програми.

4. Як зміниться діаграма, якщо використовуватиме методи підвищення надійності роботи.

Література [1,…,10]

Лабораторна робота № 11

“ Диаграмма компонентов системи підтримки туристичних подорожей ”.

Мета роботи.
Набути навичок розробки діаграми компонентів

Теоретичні положення.
Погляд на компонент як постачальника сервісів визначається двома основними характеристиками компонентів, допускають їх повторне використання.
1. Компонент – це програмний об'єкт, що незалежно виконується. Вихідний код компонента може бути недоступний, тому компонент не компілюється спільно з іншими компонентами системи.
2. Компоненти оголошують свій інтерфейс та всі взаємодії з ними здійснюються за його допомогою. Інтерфейс компонента описується в термінах параметризованих операцій, а внутрішній стан компонента завжди прихований.
Компоненти визначаються через інтерфейси. Найчастіше компоненти можна описати як двох взаємозалежних інтерфейсів:
• Інтерфейс постачальника послуг, який визначає послуги, що надаються компонентом.
• Інтерфейс запитів, який визначає, які послуги доступні компоненту із системи, що використовує цей компонент.

Компоненти можуть існувати на різних рівнях абстракції – від простої бібліотечної підпрограми до додатків, таких як Microsoft Excel. Прийнято виділяти п'ять рівнів абстракції компонентів.

1. Функціональна абстракція. Компонент реалізує окрему функцію, наприклад, математичну. Власне, інтерфейсом постачальника сервісів є сама функція.
2. Безсистемне угруповання. В даному випадку компонент – це набір слабо пов'язаних між собою програмних об'єктів та підпрограм, наприклад оголошень даних, функцій тощо. Інтерфейс постачальника послуг складається з назв всіх об'єктів у групуванні.
3. Абстракції даних. Компонент є абстракцією даних або класом, описаним об'єктно-орієнтованою мовою. Інтерфейс постачальника сервісів складається з методів (операцій), що забезпечують створення, зміну та отримання доступу до абстракції даних.
4. Абстракції кластерів. Тут компонент – це група пов'язаних класів, які працюють спільно. Такі компоненти іноді називають структурою. Інтерфейс постачальника сервісів є композицією всіх інтерфейсів об'єктів, що становлять структуру.
5. Системні абстракції. Компонент є повністю автономною системою. Повторне використання абстракцій системного рівня іноді називають повторним використанням комерційних продуктів. Інтерфейсом постачальника сервісів на цьому рівні є так званий програмний інтерфейс додатків (Application Programming Interface – API), який надає доступ до системних команд та методів. Підхід покомпонентної розробки систем можна інтегрувати в загальний процес створення ПЗ шляхом додавання спеціальних етапів, на яких відбираються та адаптуються компоненти, що повторно використовуються. Проектувальники системи розробляють системну архітектуру високому рівні абстракції, складаючи специфікації системних компонентів. Надалі ці специфікації використовуються для пошуку компонентів, що повторно використовуються. Вони включаються до системи або на рівні системної архітектури, або на нижчих деталізованих рівнях.

Завдання до роботи.
1. Розробка додаткових функціональних вимог.
2. Розробка нової діаграми компонентів

Способ обробки результатів.

Необхідно придумати додатковий функціонал до існуючого проекту, пояснити його доцільність щодо наявного функціоналу, та розробити діаграму відповідно до повного функціонала.

Домашне завдання.
Підберіть найбільш підходящий фреймворк для реалізації додаткового введеного функціоналу

Рекомендації щодо оформлення звіту з роботи.
Оформлення у суворій відповідності до [10].

Контрольні запитання за темою роботи.
1. Як зміниться діаграма, якщо використовувати архітектуру мікросервісів.
2.Як зміниться діаграма, якщо використовувати мікрофронтенд в рамках мікросервісної архітектури.
3. Як зміниться діаграма, якщо використовувати вбудоване тестування під час використання програми.
4. Як зміниться діаграма, якщо використовуватиме методи підвищення надійності роботи.

Література [1,…,10]

Лабораторна робота № 12

“ Диаграмма компонентов системи оренди автомобілів ”.

Мета роботи.
Набути навичок розробки діаграми компонентів

Теоретичні положення.
Погляд на компонент як постачальника сервісів визначається двома основними характеристиками компонентів, допускають їх повторне використання.
1. Компонент – це програмний об'єкт, що незалежно виконується. Вихідний код компонента може бути недоступний, тому компонент не компілюється спільно з іншими компонентами системи.
2. Компоненти оголошують свій інтерфейс та всі взаємодії з ними здійснюються за його допомогою. Інтерфейс компонента описується в термінах параметризованих

операцій, а внутрішній стан компонента завжди прихований.

Компоненти визначаються через інтерфейси. Найчастіше компоненти можна описати як двох взаємозалежних інтерфейсів:

- Інтерфейс постачальника послуг, який визначає послуги, що надаються компонентом.
- Інтерфейс запитів, який визначає, які послуги доступні компоненту із системи, що використовує цей компонент.

Компоненти можуть існувати на різних рівнях абстракції – від простої бібліотечної підпрограми до додатків, таких як Microsoft Excel. Прийнято виділяти п'ять рівнів абстракції компонентів.

1. Функціональна абстракція. Компонент реалізує окрему функцію, наприклад, математичну. Власне, інтерфейсом постачальника сервісів є сама функція.
2. Безсистемне угруповання. В даному випадку компонент – це набір слабо пов'язаних між собою програмних об'єктів та підпрограм, наприклад оголошень даних, функцій тощо. Інтерфейс постачальника послуг складається з назв всіх об'єктів у групуванні.
3. Абстракції даних. Компонент є абстракцією даних або класом, описаним об'єктно-орієнтованою мовою. Інтерфейс постачальника сервісів складається з методів (операцій), що забезпечують створення, зміну та отримання доступу до абстракції даних.
4. Абстракції кластерів. Тут компонент – це група пов'язаних класів, які працюють спільно. Такі компоненти іноді називають структурою. Інтерфейс постачальника сервісів є композицією всіх інтерфейсів об'єктів, що становлять структуру.
5. Системні абстракції. Компонент є повністю автономною системою. Повторне використання абстракцій системного рівня іноді називають повторним використанням комерційних продуктів. Інтерфейсом постачальника сервісів на цьому рівні є так званий програмний інтерфейс додатків (Application Programming Interface – API), який надає доступ до системних команд та методів. Підхід покомпонентної розробки систем можна інтегрувати в загальний процес створення ПЗ шляхом додавання спеціальних етапів, на яких відбираються та адаптуються компоненти, що повторно

використовуються. Проектувальники системи розробляють системну архітектуру високому рівні абстракції, складаючи специфікації системних компонентів. Надалі ці специфікації використовуються для пошуку компонентів, що повторно використовуються. Вони включаються до системи або на рівні системної архітектури, або на нижчих деталізованих рівнях.

Завдання до роботи.

1. Розробка додаткових функціональних вимог.
2. Розробка нової діаграми компонентів

Способ обробки результатів.

Необхідно придумати додатковий функціонал до існуючого проекту, пояснити його доцільність щодо наявного функціоналу, та розробити діаграму відповідно до повного функціонала.

Домашне завдання.

Підберіть найбільш підходящий фреймворк для реалізації додаткового введеного функціоналу

Рекомендації щодо оформлення звіту з роботи.

Оформлення у суворій відповідності до [10].

Контрольні запитання за темою роботи.

1. Як зміниться діаграма, якщо використовувати архітектуру мікросервісів.
2. Як зміниться діаграма, якщо використовувати мікрофронтенд в рамках мікросервісної архітектури.
3. Як зміниться діаграма, якщо використовувати вбудоване тестування під час використання програми.
4. Як зміниться діаграма, якщо використовуватиме методи підвищення надійності роботи.

Література [1,…,10]

Лабораторна робота № 13

“ Диаграмма компонентов системи управління готелем ”.

Мета роботи.

Набути навичок розробки діаграми компонентів

Теоретичні положення.

Погляд на компонент як постачальника сервісів визначається двома основними характеристиками компонентів, допускають їх повторне використання.

1. Компонент – це програмний об'єкт, що незалежно виконується. Вихідний код компонента може бути недоступний, тому компонент не компілюється спільно з іншими компонентами системи.

2. Компоненти оголошують свій інтерфейс та всі взаємодії з ними здійснюються за його допомогою. Інтерфейс компонента описується в термінах параметризованих операцій, а внутрішній стан компонента завжди прихований.

Компоненти визначаються через інтерфейси. Найчастіше компоненти можна описати як двох взаємозалежних інтерфейсів:

• Інтерфейс постачальника послуг, який визначає послуги, що надаються компонентом.

• Інтерфейс запитів, який визначає, які послуги доступні компоненту із системи, що використовує цей компонент.

Компоненти можуть існувати на різних рівнях абстракції – від простої бібліотечної підпрограми до додатків, таких як Microsoft Excel. Прийнято виділяти п'ять рівнів абстракції компонентів.

1. Функціональна абстракція. Компонент реалізує окрему функцію, наприклад, математичну. Власне, інтерфейсом постачальника сервісів є сама функція.

2. Безсистемне угруповання. В даному випадку компонент – це набір слабо пов'язаних між собою програмних об'єктів та підпрограм, наприклад оголошень даних, функцій тощо. Інтерфейс постачальника послуг складається з назв всіх об'єктів у групуванні.

3. Абстракції даних. Компонент є абстракцією даних або класом, описаним об'єктно-орієнтованою мовою. Інтерфейс постачальника сервісів складається з методів (операцій), що забезпечують створення, зміну та отримання доступу до абстракції даних.

4. Абстракції кластерів. Тут компонент – це група пов'язаних класів, які працюють спільно. Такі компоненти іноді

називають структурою. Інтерфейс постачальника сервісів є композицією всіх інтерфейсів об'єктів, що становлять структуру.

5. Системні абстракції. Компонент є повністю автономною системою. Повторне використання абстракцій системного рівня іноді називають повторним використанням комерційних продуктів. Інтерфейсом постачальника сервісів на цьому рівні є так званий програмний інтерфейс додатків (Application Programming Interface – API), який надає доступ до системних команд та методів. Підхід покомпонентної розробки систем можна інтегрувати в загальний процес створення ПЗ шляхом додавання спеціальних етапів, на яких відбираються та адаптуються компоненти, що повторно використовуються. Проектувальники системи розробляють системну архітектуру високому рівні абстракції, складаючи специфікації системних компонентів. Надалі ці специфікації використовуються для пошуку компонентів, що повторно використовуються. Вони включаються до системи або на рівні системної архітектури, або на нижчих деталізованих рівнях.

Завдання до роботи.

1. Розробка додаткових функціональних вимог.
2. Розробка нової діаграми компонентів

Способ обробки результатів.

Необхідно придумати додатковий функціонал до існуючого проекту, пояснити його доцільність щодо наявного функціоналу, та розробити діаграму відповідно до повного функціонала.

Домашне завдання.

Підберіть найбільш підходящий фреймворк для реалізації додаткового введеного функціоналу

Рекомендації щодо оформлення звіту з роботи.

Оформлення у суворій відповідності до [10].

Контрольні запитання за темою роботи.

1. Як зміниться діаграма, якщо використовувати архітектуру мікросервісів.
2. Як зміниться діаграма, якщо використовувати мікрофронтенд в рамках мікросервісної архітектури.
3. Як зміниться діаграма, якщо використовувати вбудоване тестування під час використання програми.

4. Як зміниться діаграма, якщо використовуватиме методи підвищення надійності роботи.

Література [1,…,10]

Лабораторна робота № 14

“ Диаграмма классов системи підтримки туристичних подорожей ”.

Мета роботи.

Набути навичок розробки діаграми класів

Теоретичні положення.

Клас (class) - категорія речей, які мають загальні атрибути та операції.

Класи - це будівельні блоки будь-об'єктно-орієнтованої системи. Вони являють собою опис сукупності об'єктів із загальними атрибутами, операціями, відносинами і семантикою. При проектуванні об'єктно-орієнтованих систем діаграми класів обов'язкові.

Діаграма класів - це набір статичних, декларативних елементів моделі. Діаграма класів можуть застосовуватися і при прямому проектуванні, тобто в процесі розробки нової системи, і при зворотному проектуванні - описі існуючих і використовуваних систем. Інформація з діаграми класів безпосередньо відображається в вихідний код програми - в більшості існуючих інструментів UML-моделювання можлива кодогенерацію для певної мови програмування (зазвичай Java або C ++). Таким чином, діаграма класів - кінцевий результат проектування і відправна точка процесу розробки.

Завдання до роботи.

1. Розробка додаткових функціональних вимог.
2. Розробка нової діаграми класів.

Способ обробки результатів.

Необхідно придумати додатковий функціонал до існуючого проекту, пояснити його доцільність щодо наявного функціоналу, та розробити діаграму відповідно до повного функціонала.

Домашне завдання.

Підберіть найбільш підходящий фреймворк для реалізації додаткового введеного функціоналу

Рекомендації щодо оформлення звіту з роботи.
Оформлення у суворій відповідності до [10].

Контрольні запитання за темою роботи.

1. Як зміниться діаграма, якщо використовувати архітектуру мікросервісів.
2. Як зміниться діаграма, якщо використовувати мікрофронтенд в рамках мікросервісної архітектури.
3. Як зміниться діаграма, якщо використовувати вбудоване тестування під час використання програми.
4. Як зміниться діаграма, якщо використовуватиме методи підвищення надійності роботи.

Література [1,…,10]

Лабораторна робота № 15

“ Диаграмма классов системи управління готелем ”.

Мета роботи.
Набути навичок розробки діаграми класів

Теоретичні положення.

Клас (class) - категорія речей, які мають загальні атрибути та операції.

Класи - це будівельні блоки будь-об'єктно-орієнтованої системи. Вони являють собою опис сукупності об'єктів із загальними атрибутами, операціями, відносинами і семантикою. При проектуванні об'єктно-орієнтованих систем діаграми класів обов'язкові.

Діаграма класів - це набір статичних, декларативних елементів моделі. Діаграма класів можуть застосовуватися і при прямому проектуванні, тобто в процесі розробки нової системи, і при зворотному проектуванні - описі існуючих і використовуваних систем. Інформація з діаграми класів безпосередньо відображається в вихідний код програми - в більшості існуючих інструментів UML-моделювання можлива кодогенерацію для певної мови програмування (зазвичай Java або C ++). Таким чином, діаграма класів - кінцевий результат проектування і відправна точка процесу розробки.

Завдання до роботи.

1. Розробка додаткових функціональних вимог.

2. Розробка нової діаграми класів.

Способ обробки результатів.

Необхідно придумати додатковий функціонал до існуючого проекту, пояснити його доцільність щодо наявного функціоналу, та розробити діаграму відповідно до повного функціонала.

Домашне завдання.

Підберіть найбільш підходящий фреймворк для реалізації додаткового введеного функціоналу

Рекомендації щодо оформлення звіту з роботи.

Оформлення у суворій відповідності до [10].

Контрольні запитання за темою роботи.

1. Як зміниться діаграма, якщо використовувати архітектуру мікросервісів.
2. Як зміниться діаграма, якщо використовувати мікрофронтенд в рамках мікросервісної архітектури.
3. Як зміниться діаграма, якщо використовувати вбудоване тестування під час використання програми.
4. Як зміниться діаграма, якщо використовуватиме методи підвищення надійності роботи.

Література [1,…,10]

Максимальна кількість балів за одну лабораторну роботу – 7 балів. Діаграми оцінюються порівняно з діаграмами у додатку.

Список літератури

1. Hyrum Wright, Tom Manshreck, Titus Winters.- Software Engineering at Google.- O'Reilly Media, 2020.-250p.
2. Mark Richards, Neal Ford.- Fundamentals of Software Architecture: An Engineering Approach.- O'Reilly Media, 2020.-500p.
3. Bhuvan Unhelkar.- Software Engineering with UML.- Auerbach Publications;CRC PRESS, 2018.-427p.
4. David Farley.- Modern Software Engineering: Doing What Works to Build Better Software Faster (Rough Cut).- Addison-Wesley Professional, 2022.-256p.
5. Ian Sommerville.- Software Engineering.- Pearson, 2015.-816p.

6. Volker Gruhn, Rüdiger Striemer.- The Essence of Software Engineering.- Springer International Publishing, 2018.-247p.
7. Unhelkar B. - Software Engineering with UML. - Auerbach Publications; 1st edition, 2017. – 406p.
8. Галчонков О.М. Конспект лекцій з дисципліни "Технології комп'ютерного проектування" розроблений для здобувачів інституту комп'ютерних систем усіх форм навчання за спеціальністю 122 - Комп'ютерні науки / Укладач: О. М. Галчонков - Одеса: НУОП, 2022. - 95с.
Регістраційний № 3894-РС-2022 від 05.12.2022
9 .Галчонков О.М. Методичні вказівки до лабораторних робіт з дисципліни «Технології комп'ютерного проектування» для здобувачів всіх форм навчання інституту комп'ютерних систем (спеціальність 122 – «Комп'ютерні науки») / Укладач: О.М.Галчонков – Одеса: НУОП, 2022. – 74 с.
Регістраційний № 3895-РС-2022 від 05.12.2022
10. Антощук С.Г., Арсірій О.О., Щербакова Г.Ю.,. Бабілунга О.Ю, Блажко О.А., Галчонков О.М., Глава М.Г., Ядрова М.В. Методичні вказівки до виконання та захисту кваліфікаційної роботи бакалавра (атестації здобувача вищої освіти першого рівня) за освітньо-професійною програмою 122 «Комп’ютерні науки» / Укл.: С.Г. Антощук, О.О. Арсірій, Г.Ю. Щербакова О.Ю. Бабілунга, О.А. Блажко, О.М. Галчонков, М.Г. Глава, М.В. Ядрова. – Одеса: Одеська політехніка, 2022. – 28 с.