**Міністерство освіти і науки України**

**Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського"**

**Факультет інформатики та обчислювальної техніки**

**Кафедра інформатики та програмної інженерії**

**Домашня контрольна робота №2**

Варіант 2

з дисципліни

«Компоненті програмної інженерії 3.»

**Виконав(ла)**

(шифр, прізвище, ім'я, по батькові)

*ІП-92 Бераудо Еліза*

**Перевірив**

(прізвище, ім'я, по батькові)

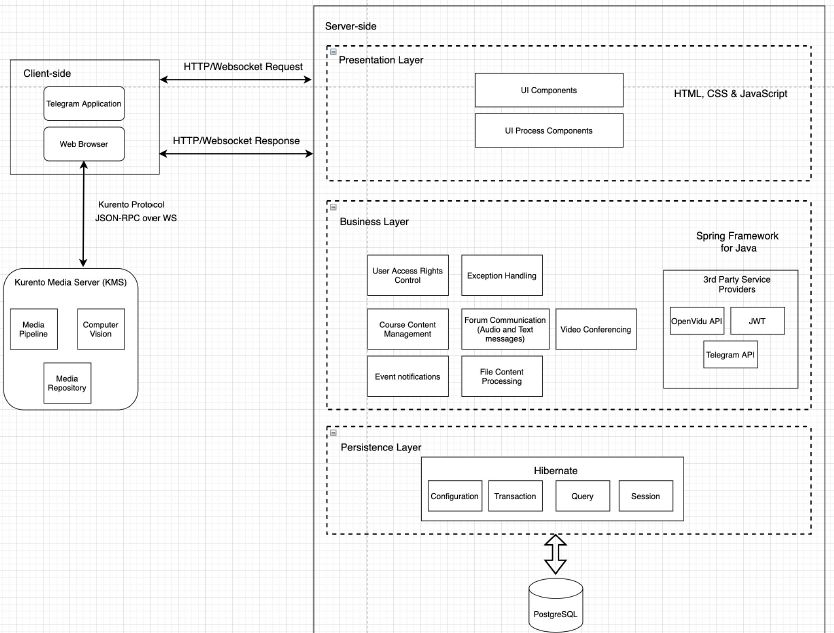
*Головченко М.М.*

Київ 2021

# Завдання

1. завдання 1

Який архітектурний паттерн використовується на зображеній на рисунку архітектурі програмного забезпечення? Описати дане програмне забезпечення.



На даній діаграммі присутній багаторівневий архітектурний паттерн. Тому, що присутні 4 рівня: представлення даних, бізнес-логіка, зберігання даних та база даних. По-перше, кожен з рівнів має певну роль і відповідальність. Наприклад, рівень представлення даних відповідає за обробку всього інтерфейсу користувача. Такий поділ завдань у багаторівневій архітектурі спрощує формування ефективних ролей та відповідальності. По-друге, шаблон багаторівневої архітектури є технічним поділом - на відміну від поділу по доменам: тобто це групи компонентів, а не домени. І, нарешті, третя ідея: кожен рівень позначається як закритий чи відкритий. Запит, переміщаючись з рівня на рівень і минаючи закритий рівень, повинен обов'язково пройти через нього - пропустити закритий рівень не можна.

Використовуэться База Даних PostgreSQL для зберігання, зміни та обробки взаємопов'язаної інформації, переважно великих обсягів

У першому рівні Persistence, на діаграммі використовується Hibernate, щоб мати взаємодію з Базою Даних.

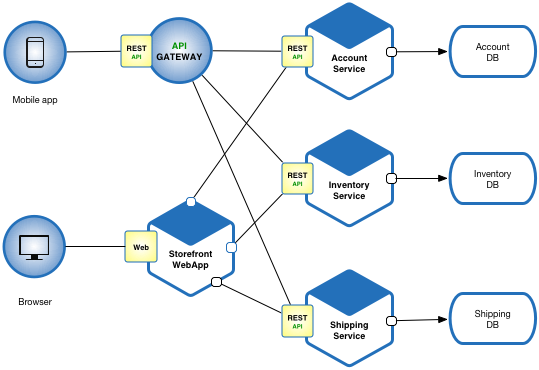
У другому Business, ми бачимо різноманітні форми для мітінгів, сповіщення повідомлень, форми коммунікацій, менеджмент, тобто управління курсів та мітингів, аксес юзера також використання різноманітних бібліотек таких як JWT та 3rd Party Services OpenVideo Api та Telegram Api.

На останньому рівні Представлення даних відбувається рендерингу сторінок на стороні сервера, що дозволяє зменшити значення затримки на нуль. Юзер має можливість отримувати сповіщення в Веб-Броузер та Телеграм. Завдяки Websocket, надається можливість використовувати можливості мітингу. Завдяки Kurento – це безкоштовний WebRTC медіасервер з відкритим вихідним кодом. Він визначає медіапотоки між користувачами, транскодує відео, робить запис.

Отже, на даній діаграммі зображено програмне забезпечення мітінгу, наприкладі, Telegram application.

1. завдання 2

Який архітектурний паттерн використовується на зображеній на рисунку архітектурі програмного забезпечення? Описати дане програмне забезпечення.

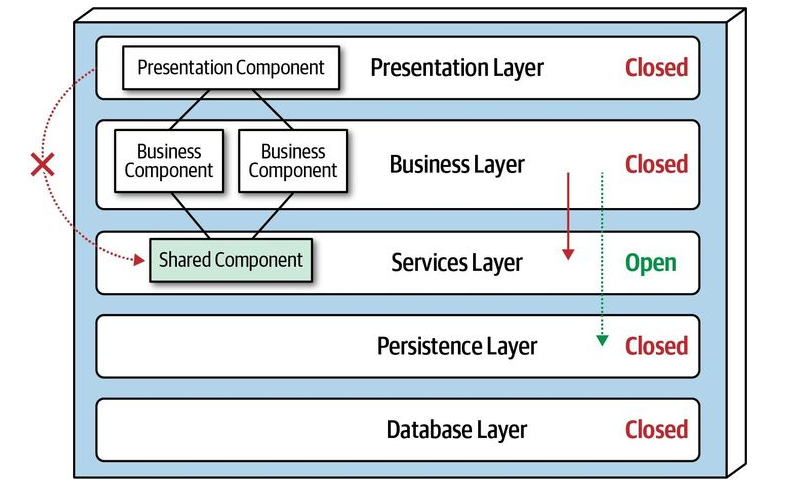


На цьому приклады зображено мікросервісна архітектура. Тому, що це важлива організація розподіленої системи з урахуванням мікросервісів та його взаємодії друг з одним і з середовищем у мережі, і навіть принципів, що спрямовують проектування архітектури, її створення та еволюцію. На діаграммі зображено, 2 точки співдіяння, 1- Веб-сервісний інтерфейс, 2- API, для смайрфонів. Перший та другий застосунок надають нам змогу для взаємодії с такими сервісами як: Account, Inventory, Shipping. На дыаграммы видно, що кожний сервыс має взасну БД.

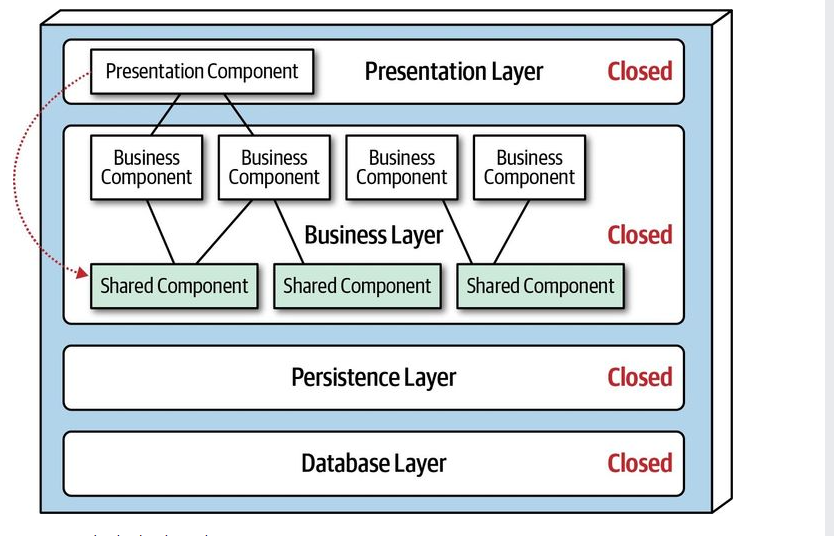
1. завдання 3

Навести приклад програмного забезпечення (опис) крім лекційного, з закритими і відкритими рівнями у багаторівневій архітектурі.

На відміну від закритого, відкритий рівень, може проходити через рівні, наприклад, з представлення даних до зберігання данних.



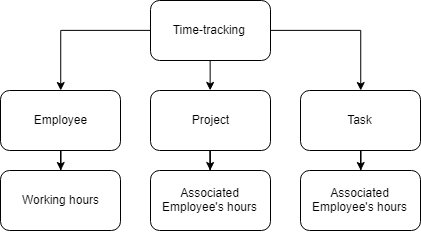
Суть закритого, що він має пройти через попередны рівні. Приклад на діаграммі знизу. Я зобразила це черними стрілочками.

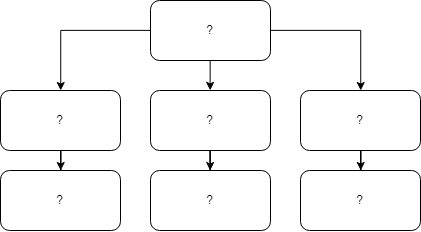


1. завдання 4

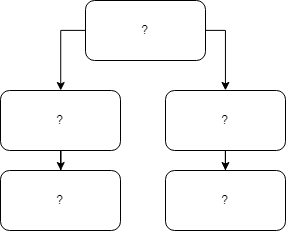
Навести приклад програмного забезпечення (опис предметної області та схема архітектури), в якому буде проявлятись антипаттерн **велика куля бруду (Big ball of mud)**

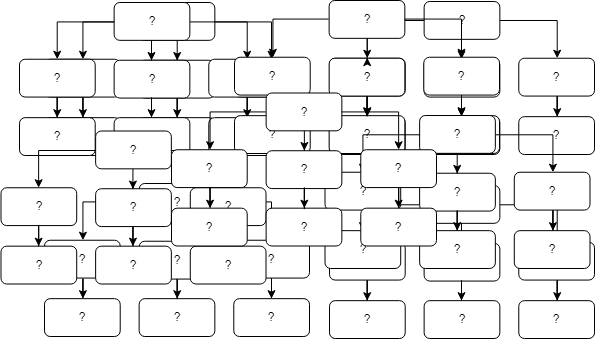
Велика куля грязі формується, ще до того як проект пояинають писати. Хочу взяти у приклад систему Time Track, яка надає можливість співробітникам реєструвати час, витрачений на викопання різноманітних завдань та проектів.



Припустим, що на початку проекту його дизайн був успішно зроблений. Після того додаються нові ідеї для реалізації проекту, зявляються нові концепції: команди, різноманітні рахунки, інтеграції

Звявляються нові поняття



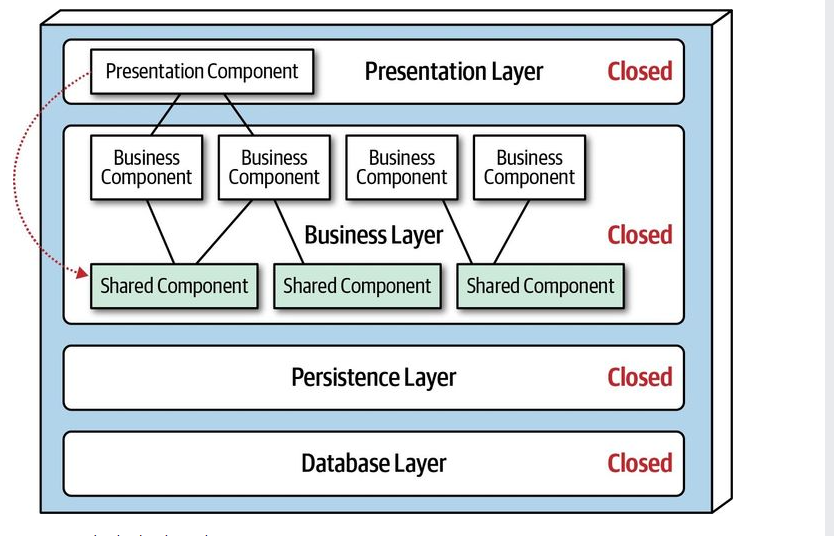
Зараз концепції продукту: команди, різноманітні рахунки, інтеграції, почали перемішуватися, обмежені контести не були визначені, терміни двозначні, мова зіпсована.

1. **Завдання 5**

Навести приклад програмного забезпечення (опис предметної області та схема архітектури), в якому буде проявлятись антипаттерн **воронки (вирви) (architecture sinkhole anti-pattern)**

Так званий sinkhole Anti pattern.

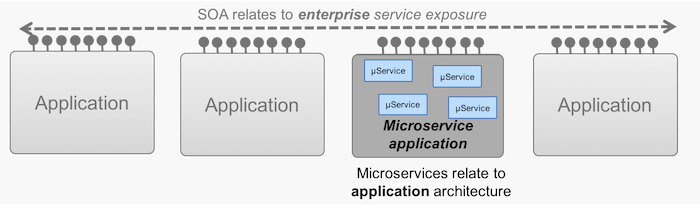
Наприклад, рівень презентації отримує запит на отримання основних даних користувача (повне ім'я, адресу тощо) і передає його на бізнес-рівень. Однак бізнес-рівень не виконує жодної бізнес-обробки. Передайте запит безпосередньо на рівень збереження. Рівень збереження просто створює просте речення SQL, запитувати дані користувача до рівня даних. Нарешті, дані повертаються на рівень презентації таким же чином, немає агрегації даних, перемикання тощо.



1. **Завдання 6**

У чому принципова різниця між архітектурними паттернами мікросервісної архітектури та сервіс-орієнтованої архітектури (SOA)?

Основна відмінність між двома підходами зводиться до сфери застосування. Простіше кажучи, сервіс-орієнтована архітектура (SOA) має корпоративну сферу, тоді як архітектура мікросервісів має область застосування.



Багато основних принципів кожного підходу стають несумісними, якщо нехтувати цією різницею. Якщо ви погодитеся на різницю в масштабах, ви можете швидко зрозуміти, що вони потенційно можуть доповнювати один одного, а не конкурувати.

Нижче наведено кілька випадків використання, коли ця відмінність вступає в гру:

Повторне використання

У SOA повторне використання інтеграцій є основною метою, а на рівні підприємства важливо прагнути до певного рівня повторного використання. Повторне використання та спільне використання компонентів в архітектурі SOA підвищує масштабованість та ефективність.

В архітектурі мікросервісів створення компонента мікросервісів, який повторно використовується під час виконання програми, призводить до залежностей, які знижують гнучкість і стійкість. Компоненти мікросервісів зазвичай вважають за краще повторно використовувати код шляхом копіювання та приймання дублювання даних, щоб покращити роз’єднання.

Зв'язок: в архітектурі мікросервісів кожна служба розробляється незалежно, з власним протоколом зв'язку. З SOA кожна служба має використовувати загальний механізм зв’язку, який називається корпоративною службовою шиною (ESB). SOA керує та координує послуги, які надає через ESB. Однак ESB може стати єдиною точкою збою для всього підприємства, і якщо одна служба сповільнюється, це може постраждати від усієї системи.

Сумісність: в інтересах простоти мікросервіси використовують полегшені протоколи обміну повідомленнями, такі як HTTP/REST (Передача репрезентаційного стану) і JMS (Служба обміну повідомленнями Java). SOA більш відкриті для різнорідних протоколів обміну повідомленнями, таких як SOAP (Протокол доступу до об’єктів), AMQP (Протокол розширеного обміну повідомленнями) і MSMQ (черга обміну повідомленнями Microsoft).

Детальність сервісу: архітектури мікросервісів складаються з високоспеціалізованих служб, кожен з яких розроблений для того, щоб дуже добре виконувати одну справу. З іншого боку, послуги, що входять до складу SOA, можуть варіюватися від невеликих спеціалізованих служб до послуг для всього підприємства.

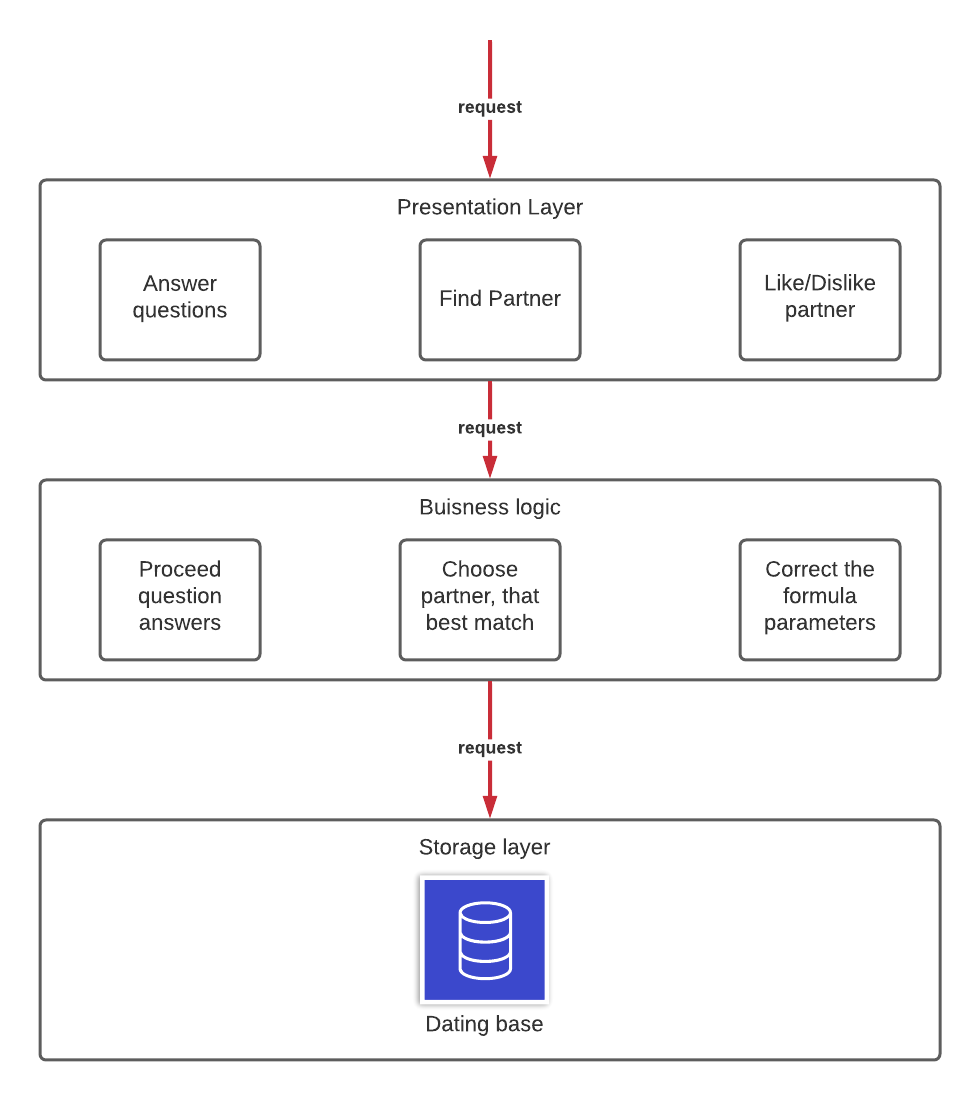
Швидкість: використовуючи переваги спільної архітектури, SOA спрощують розробку та усунення несправностей. Однак це також призводить до того, що SOA працюють повільніше, ніж архітектури мікросервісів, що мінімізує спільний доступ на користь дублювання.

Керування: природа SOA, що включає спільні ресурси, дозволяє впроваджувати загальні стандарти керування даними для всіх служб. Незалежна природа мікросервісів не забезпечує послідовне керування даними. Це забезпечує більшу гнучкість для кожної служби, що може сприяти більшій співпраці в організації.

Зберігання: SOA і мікросервіси також відрізняються з точки зору того, як розподіляються ресурси зберігання. Архітектура SOA зазвичай включає в себе один рівень зберігання даних, який спільно використовують усі служби в межах даної програми, тоді як мікросервіси виділяють сервер або базу даних для зберігання даних для будь-якої служби, яка цього потребує.

1. **Завдання 7**

**(варіант 2)** Навести приклад програмного забезпечення (опис предметної області і втілення архітектурного паттерну) для якого, є доцільним застосування багаторівневого архітектурного паттерну (крім лекційного і лабораторних).



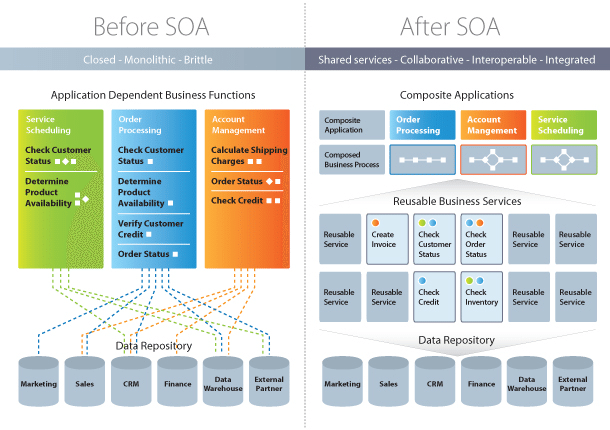
Прикладом може слугувати стартап (новий проект) з пошуку найкращої пари для людини. На рівні представлення користувач може заповнити інформацію про себе, вибрати партнера, який найкраще підходить користувачеві та відповісти на запитання: «Чи сподобався йому попередній партнер?»

На бізнес рівні, ми обробляємо відповіді користувача та створюємо коефіцієнти для рівняння пошуку, вибираємо найкращого партнера за заданим рівнянням або коригуємо зміні рівняння пошуку.

Всі зміни відображаються на рівні збереження у Базі даних «Dating base».

1. **Завдання 8**

**(варіант 2)** Навести приклад програмного забезпечення (опис предметної області і втілення архітектурного паттерну) для якого, є доцільним застосування сервіс-орієнтованого архітектурного паттерну (крім лекційного і лабораторних).

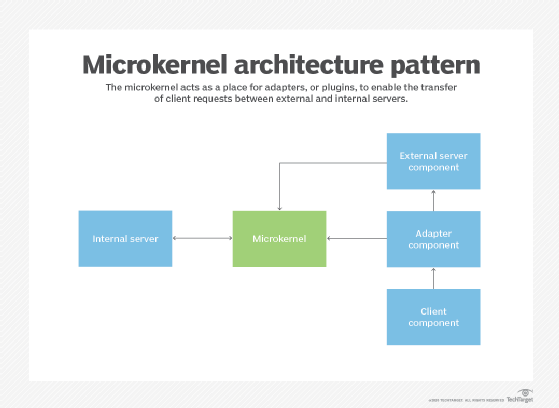


Розглянемо, як приклад інтернет магазин з продажу товарів та оцінимо якість архітектури з SOA та без цього паттерна. До прикладу, без цього патерну ми бачимо, що частина функціоналу у нас дублюється. Це може призвести до складності його підтримки. (OrderStatus функція використовується як в Account Management так і в Order Processing). Також користь цього архітектурного підходу заключається в зменшені витрати на планування та розробку ПЗ, а також в легкості масштабування застосунку (за рахунок зменшення взаємодії між клієнтом та сервером). Також через вкористання центрального інтерфейсу користувач має змогу використовувати різні мови програмування.

В нашому прикладі після завстосування патернк проектування системи ми уникнули повторення програмного коду, спростили складність системи, додали можливість легкого розширення та зменшили вартість нашого рішення в цілому.

1. **Завдання 9**

**(варіант 2)** Навести приклад програмного забезпечення (опис предметної області і втілення архітектурного паттерну) для якого, є доцільним застосування мікроядерного архітектурного паттерну (крім лекційного і лабораторних).



Прикладом мікроядерної архітектури можуть слугувати такі продукти, як Google Chrome, Firefox, Edge, Intelij Idea та мій улюблений VSCode. Тобто завдяки мікроядерній архітектурі, користувачеві надається можливість розширити функціонал або додати відсутній функціонал використовуючи API головної програми. Ця гнучкість стала головною перевагою VSCode, оскільки кожен може написати розширення та додати відсутній потрібний функціонал, що може перетворити текстовий редактор в повноцінну конкуруючу середу розробки.

Говорячи про переваги мікроядерної архітектури, то можна зазначити такі:

* Висока гнучкість
* Простота розгортання
* Легко тестується

1. **Завдання 10**

**(варіант 2)** Навести приклад програмного забезпечення (опис предметної області і втілення архітектурного паттерну) для якого, є доцільним застосування архітектурного паттерну Model-view-controller (крім лекційного і лабораторних).



Прикладом MVC патерну може слугувати відеогра Inside (може бути будь-яка гра. Вибір впав через те, що автор її не давно проходив). В ній користувач має п’ять основних дій: «рух вправо», «рух вліво», «рух вниз», «рух вверх» та взаємодія з предметом біля персонажа. Модель це позиція персонажа на карті, його стан (скільки часу він провів під водою) та позиція ворогів (собак, дівчинки з довгим волоссям або переслідувачів). View відображає модель у вигляді зображення на екрані або звуків. Коли користувач натискає кнопку, представлення відсилає кнопку до контролера, який перевіряє різні умови(фізику, стан світу) та оновлює відповідно модель. Модель посилає сигнал до View, що вона була змінена і View забирає новий стан та оновлює зображення на екрані. Наприклад, так як ми забули виманити охорону і зробили крок вліво нас піймали патрульні.

Изображение выглядит как темный, ночь

Автоматически созданное описание

Критерії оцінювання

Термін виконання ДМКР2 28.12.2021 включно максимальний бал дорівнює – 10. Кожне питання оцінюється в 1 бал.

Після 28.12.2021 ДМКР2 – НЕ приймаються.