Міністерство освіти і науки України

Відокремлений структурний підрозділ

«Запорізький електротехнічний фаховий коледж

Національного університету «Запорізька політехніка»

## ТЕХНОЛОГІЧНА ПРАКТИКА

### Звіт з практики

121.49.29.01 ЗП

Керівники практики

від коледжу Олеся ТКАЧУК

від підприємства Олександр КОРОТКИЙ

Студент групи РПЗ 19 2/9 Іван ЩЕДРОВСЬКИЙ

Місце проходження практики ВСП «ЗЕФК НУ «Запорізька політехніка»

2023

**ЗМІСТ**

[1 ОПИС ПІДПРИЄМСТВА 3](#_Toc127800371)

[1.1 Коротка характеристика підприємства 3](#_Toc127800372)

[1.2 Схема структури підприємства 6](#_Toc127800373)

[1.3 Опис структури комп’ютерної мережі підприємства 7](#_Toc127800374)

[1.4 Аналіз апаратного і програмного забезпечення, що використовується на підприємстві 8](#_Toc127800375)

[2 ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ, ЩО ПІДЛЯГАЄ АВТОМАТИЗАЦІЇ 10](#_Toc127800377)

[3 СТВОРЕННЯ ПОСТАНОВИ ЗАВДАННЯ ДО ДИПЛОМНОЇ РОБОТИ 10](#_Toc127800378)

[4 ПРОГРАМУВАННЯ 22](#_Toc127800379)

[4.1 Проектування інтерфейсу користувача 22](#_Toc127800380)

[4.2 Створення головного модуля програми 23](#_Toc127800381)

[ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ 23](#_Toc127800382)

[ДОДАТОК А 28](#_Toc127800383)

[Текст головного модуля програми 28](#_Toc127800384)

**1 ОПИС ПІДПРИЄМСТВА**

**1.1 Коротка характеристика підприємства**

Відокремлений структурний підрозділ Запорізький електротехнічний фаховий коледж Національного університету «Запорізька політехніка» - навчальний заклад України I-II рівнів акредитації Придніпровського промислового регіону.

Історичний шлях розвитку навчального закладу розпочинається з 1935 року, як Запорізького алюмінієвого технікуму.

За роки існування в коледжі отримали путівку в життя понад 30 тисяч молодих фахівців.

Запорізький електротехнічний коледж [Запорізького національного технічного університету](http://uk.wikipedia.org/wiki/%D0%97%D0%B0%D0%BF%D0%BE%D1%80%D1%96%D0%B7%D1%8C%D0%BA%D0%B8%D0%B9_%D0%BD%D0%B0%D1%86%D1%96%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D0%B8%D0%B9_%D1%82%D0%B5%D1%85%D0%BD%D1%96%D1%87%D0%BD%D0%B8%D0%B9_%D1%83%D0%BD%D1%96%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D1%82%D0%B5%D1%82) забезпечує різнорівневу підготовку, перепідготовку молодших спеціалістів і бакалаврів, підвищення кваліфікації, професійну та курсову підготовку фахівців для реалізації безперервної ступеневої освіти на Україні, що відповідає основним напрямкам Законів „Про освіту”, „Про вищу освіту”, концепціям, викладеним в національній програмі „Освіта. Україна XXI століття”, положенням Болонського процесу навчання.

Сьогодні коледж здійснює підготовку фахівців за освітньо - кваліфікаційним рівнем молодший спеціаліст на денній та заочній формах навчання за державним замовленням та на контрактній основі з таких спеціальностей: Економіка, [Електроніка](http://zetk.com.ua/electronics/)**,** [Галузеве машинобудування](http://zetk.com.ua/mechanical-engineering/)**,** [Електроенергетика, електротехніка та електромеханіка](http://zetk.com.ua/electronic-engineering/)**,** [Інженерія програмного забезпечення](http://zetk.com.ua/information-technology/)**,** Дизайн.

За бажанням студенти можуть отримати другу вищу освіту та дипломи державного зразка на відділенні післядипломної освіти, а також здобути додаткові знання і вміння в центрі курсової перепідготовки та підвищення кваліфікації з економіки, бухгалтерського обліку, комп'ютерних технологій.

Виконуючи соціальні замовлення регіону, коледж має всі необхідні умови для якісного підвищення кваліфікації, спеціалізації фахівців у галузі знань "Економіка і підприємництво", а саме: акредитовані спеціальності з підготовки молодших спеціалістів; матеріальну базу, що відповідає вимогам МОН України; високопрофесійні викладацькі кадри з акредитованих спеціальностей, які мають особистий практичний досвід роботи з економіки і підприємництва, досвід співпраці з провідними організаціями й підприємствами регіону щодо перепідготовки фахівців; замовлення на розширення профілю (підвищення кваліфікації) і спеціалізації від Запорізького обласного центру зайнятості; врахування при здійсненні освітньої діяльності динаміки змін у потребах підприємств; необхідне навчально - методичне забезпечення.

На базі обчислювального центру коледжу працює Центр інформаційних технологій («ІНТЕХ»), який забезпечує виконання завдань усіх відділень з питань використання обчислювальної техніки та впровадження сучасних інформаційних технологій у навчально-виховний процес. Також «ІНТЕХ» надає освітні послуги підприємствам Запоріжжя та іншим замовникам з вирішення виробничих завдань щодо підготовки та перепідготовки кадрів, з розробки конструкторсько-технологічної документації з використанням комп'ютерної техніки, моделювання та програмування, виконання інженерних розрахунків, оснащення і введення в експлуатацію автоматизованої бухгалтерської та планово-економічної звітності. На базі Центру інформаційних технологій «ІНТЕХ», проводяться науково-методичні семінари за загальною тематикою «Комп'ютерні технології: нові рішення».

З 2003 року коледж здійснює підготовку майбутніх студентів на підготовчих курсах до вступу у вищі навчальні заклади України. Більшість слухачів підготовчих курсів стають студентами коледжу. Саме тому рейтинг підготовчих курсів щорічно зростає.

Сьогодні у коледжі на денному відділенні навчається більше 1000 студентів, на заочному - 300 студентів. Щорічно 100 студентів здобувають знання на відділенні післядипломної освіти, понад 500 слухачів закінчують курси перепідготовки.

Навчально-виховний процес забезпечує найкращий у регіоні викладацький склад – висококваліфіковані викладачі коледжу та інших навчальних закладів III–IV рівнів акредитації, які мають наукові ступені та вчені звання. Очолює педагогічний колектив директор Рябенко Є.М.

Заняття проводяться з використанням інноваційних технологій, сучасних дидактичних засобів навчання.

Впродовж останніх років здійснюється інтенсивна комп'ютеризація та інформатизація навчального процесу. В комп'ютерних класах проводяться практичні заняття, виконуються розрахунки з курсового та дипломного проектування, а також з фундаментальних і спеціальних дисциплін. У період навчання передбачається оволодіння прикладними комп'ютерними програмами за фахом, а також придбання навичок з математичного моделювання, з прийняття конструкторських і технологічних рішень.

Коледж має достатню науково - матеріальну базу: навчальні корпуси, у яких розміщено 50 спеціалізованих кабінетів і лабораторій, обладнаних сучасним устаткуванням, навчально- виробничі майстерні, 10 комп'ютерних класів, бібліотеку з читальною залою, їдальню, дві спортивні зали, тренажерні зали для заняття атлетичною гімнастикою, музеї бойової слави та історії навчального закладу, медичний пункт, гуртожиток.

Студентами і викладачами коледжу широко використовується мережа Internet, до якої підключено 166 персональних комп'ютери, а у локальній мережі працюють 166 ПЕОМ.

У навчальному закладі діє студентське самоврядування, працюють гуртки художньої самодіяльності, технічної творчості, спортивні секції.

Зберігаючи найкращі традиції попередніх поколінь, студенти мають змогу набувати необхідного виробничого досвіду на найкращих промислових підприємствах нашого міста.

В коледжі діє ступенева освіта, яка дає можливість випускникам продовжити навчання з обраної спеціальності за скороченим терміном у Запорізькому національному технічному університеті, а також в інших вищих навчальних закладах III -IV рівнів акредитації.

Моніторинг ринку праці показує, що в Запорізькій області, яка є однією із промислово розвинених в Україні, випускники коледжу потрібні на провідних підприємствах міста, з якими коледж плідно співпрацює, зокрема з такими, як: АТ «Мотор Січ», ПАТ «Запоріжсталь», ПрАТ «Запоріжтрансформатор», ПрАТ «Дніпроспецсталь», ПАТ «Запорізький завод важкого машинобудування», ДП ЗДАРЗ «МіГремонт», ПАТ «Запоріжжяобленерго», ПАТ «Укрграфіт», ТОВ «Елемент – Перетворювач», ТОВ «ЗТМК», ПрАТ «Плутон», ТОВ «ЗАЗОСНАСТКА», ТОВ «ЕЛІЗ» тощо.

Наш коледж розташований на одній з найкрасивіших вулиць центру міста. Усі іногородні студенти забезпечуються гуртожитком.

**1.2 Схема структури підприємства**

На рисунку 1.1 зображена схема структури підприємства

|  |
| --- |
| Директор |

|  |  |  |
| --- | --- | --- |
| Заступник директора з НР |  | Заступник директора з ВР |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Завідувачі відділень |  | Головний бухгалтер |  | Навчальна частина |  | Приймальна комісія |  | Класні керівники |

Рисунок 1.1 – Схема структури підприємства

Директор виконує функції контролю за фінансами підприємства, освітньою діяльністю коледжу і приймає важливі рішення при виникненні проблемних ситуацій в діяльності підприємства.

Заступник директора з навчальної роботи виконує оперативне управління відділеннями та приймальною комісією.

Заступник директора з виховної роботи контролює проведення в коледжі виховних позакласних заходів, координує роботу класних керівників.

Бухгалтерія веде облік фінансів підприємства, проводить розрахунки з контрактними студентами, податковий облік та виконує розрахунок і виплату заробітної платні – поєднує функції бухгалтерії і відділу праці та заробітної платні.

Навчальна частина складає розклад занять, контролює роботу з навчальними журналами, координує заміни викладачів тощо.

Приймальна комісія проводить профорієнтаційну роботу, організує проведення днів відкритих дверей, працює з абітурієнтами.

Класні керівники координують роботу класних груп, слідкують за успішністю та відвідуванням занять.

**1.3 Опис структури комп’ютерної мережі підприємства**

Існуюча комп'ютерна мережа районного відділу освіти складається з   
2 незалежних мереж ЛКМ 1, ЛКМ 2.

Розташування мереж приведене в таблиці 1.1.

Таблиця 1- Розташування існуючих мереж

|  |  |
| --- | --- |
| Найменування мережі | Розташування |
| ЛКМ 1 | Бухгалтерія |
| ЛКМ 2 | Комп’ютерні класи |

**1.4 Аналіз апаратного і програмного забезпечення, що використовується на підприємстві**

Апаратні засоби бухгалтерії.

Робочі станції: центральний процесор Intel Pentium 3.0 ГГц, об'єм оперативного запам'ятовуючого пристрою 1Гб, об'єм дискової підсистеми 160 Гб, мережний адаптер 100 МБіт/с.

Сервери: 1- центральний процесор Intel Celeron 2,8 ГГц, об'єм оперативного запам'ятовуючого пристрою 1,5Гб, об'єм дискової підсистеми 186Гб, мережний адаптер 100 МБіт/ с;

2 - центральний процесор AMDSempron 2.11ГГц, об'єм оперативного запам'ятовуючого пристрою 1Гб, об'єм дискової підсистеми 80 Гб, мережний адаптер 100 МБіт/с.

Передаюче середовище: кабель, неекранована вита пара категорії три, з'єднувачі RJ-45.

В бухгалтерії використовується таке програмне забезпечення:

* 1С бухгалтерія 8 – для ведення бухгалтерського обліку на підприємстві;
* Бест-Звіт – для подання звітів до органів Державної статистики, Пенсійного фонду, податкової інспекції.

Апаратні засоби комп’ютерних класів.

Робочі станції: центральний процесор Intel Celeron 2,5 ГГц, об'єм оперативного запам'ятовуючого пристрою 1 Гб, об'єм дискової підсистеми 186Гб, мережний адаптер 100 МБіт/с.

Сервер: центральний процесор AMDSempron 2.71 ГГц, об'єм оперативного запам'ятовуючого пристрою 2Гб, об'єм дискової підсистеми 186Гб, мережний адаптер 100 МБіт/с.

Передаюче середовище: кабель неекранована вита пара категорії три, з'єднувачі RJ-45

В комп’ютерних класах використовується таке програмне забезпечення:

# MS Office – пакет програм, що включають текстовий редактор, табличний процесор, систему керування базами даних, електронні презентації та ін.;

* MS Visual Studio 10;
* Borland Delphi;
* Adobe Photoshop;
* CorelDraw;
* Web-server Apache.

**2 ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ, ЩО ПІДЛЯГАЄ АВТОМАТИЗАЦІЇ**

Предметна область — це частина реального світу, що розглядається в межах певного контексту.

Тематикою моєї роботи є створення сервісу онлайн продажів.

Сервіс онлайн продажів - сервіс в інтернеті, де відбувається прямий продаж товарів споживачеві (юридичній або фізичній особі). При цьому розміщення споживацької інформації, замовлення товару і угода відбуваються там само, всередині мережі (на сайті інтернет-магазину).

Розробка програмного забезпечення — це процес, спрямований на створення та підтримку працездатності, якості та надійності програмного забезпечення, використовуючи технології, методологію та практики з інформатики, керування проєктами, математики, інженерії та інших областей знання.

Електронна комерція — це сфера цифрової економіки, що включає всі фінансові та торгові транзакції, які проводяться за допомогою комп'ютерних мереж, та бізнес-процеси, пов'язані з проведенням цих транзакцій.

Елементами сервісу онлайн продаж є: товар, користувач, каталог, кошик, особистий кабінет

Каталог — список, перелік товарів, складений у певному порядку, щоб полегшити їхній пошук.

Товар — продукт природи і людської праці або тільки людської праці у матеріальній і нематеріальній субстанції та у формі послуг, який завдяки своїм властивостям здатний задовольняти наявні чи передбачувані суспільні потреби.

Кошик - список, перелік товарів які користувач планує купити. Створений щоб полегшити перегляд, видалення або редагування кількості товарів

Особистий кабінет - це особливий розділ, який дозволяє користувачу отримати доступ до даних про стан та деталей замовлення.

Архітектура програмного забезпечення — спосіб структурування програмної або обчислювальної системи, абстракція елементів системи на певній фазі її роботи. Система може складатись з кількох рівнів абстракції і мати багато фаз роботи, кожна з яких може мати окрему архітектуру.

Архітектура клієнт-сервер є одним із архітектурних шаблонів програмного забезпечення та є домінуючою концепцією у створенні розподілених мережних застосунків і передбачає взаємодію та обмін даними між ними.

Сервіс — це обслуговування населення, забезпечення його побутових потреб.

Існує два загальних абстрактних понять Архітектури - перший пов’язаний з розбиттям системи на найбільш значимі складові частини; в другому випадку маються на увазі деякі конструктивні рішення, котрі після їх прийняття важко піддаються внесенню змін. Також, є розуміння того, що існує більше одного способу описання архітектури і ступінь важливості кожного з них змінюється з плином життєвого циклу системи

В більшості корпоративних додатків відслідковується та чи інша форма архітектурного «розшарування»

Концепція шарів(або рівнів) – одна з загально використовуваних моделей, використовуваних розробниками програмного забезпечення для розділення складних систем на більш прості частини

На рисунку 1.1 представлений приклад розділення додатку на шари



Рисунок 1.1 – Приклад розділення додатку на шари

Описуючи систему в термінах архітектурних шарів, зручно сприймати підсистеми, з яких вона складається у вигляді «багатошарового пирога».

Шар більш високого рівня користується послугами, що надає нижній шар, але той не знає про існування сусіднього верхнього рівня. Більше того, зазвичай кожен проміжний шар приховує нижній шар від верхнього.

Розділення системи на шари надає цілий ряд переваг:

- окремий шар можна сприймати як єдине самодостатнє ціле, не піклуючись про наявність інших шарів;

- можна обрати альтернативну реалізацію базових шарів;

- залежність між шарами зводиться до мінімуму;

- кожен шар є кандидатом на стандартизацію;

- якісно створений шар може слугувати основою для декількох різних шарів більш високого рівня.

Схемі розшарування властиві певні недоліки: шари здатні вдало інкапсулювати багато, але не все; модифікація одного шару одночасно пов’язана з потребою внесення каскадних змін в інші шари.

Другим недоліком є те, що наявність додаткових шарів знижує продуктивність системи. При переході від шару до шару сутності зазвичай піддаються трансформації з одного представлення в інше.

Не зважаючи на це, інкапсуляція нижче розташованих шарів дозволяє досягнути істотних переваг. Наприклад, оптимізація шару транзакцій зазвичай приводить до підвищення продуктивності всіх шарів, що розташовані вище.

Поняття шару набуло очевидної значущості в середині 1990-х років з появою архітектури клієнт-сервер. Це були системи з двома шарами, клієнт відповідав за роботу інтерфейсу користувача і виконання коду додатка, а роль сервера виконувала СКБД

Двошарова архітектура програмного забезпечення зазвичай відповідає моделі “товстого” клієнта. В такій моделі серверні компоненти системи відповідають, головним чином, за організацію зберігання і доступу до даних, а всі або більшість функцій прикладної обробки даних виконуються на стороні клієнтської частини.

Шари прикладних рішень і засоби підтримки виконання програм прикладного шару, що входять в системний шар функціонують на робочій станції, а засоби організації зберігання і доступу до даних - здебільшого на сервері.

На рисунку 1.2 представлено приклад двошарової архітектури



Рисунок 1.2 - Приклад двошарової архітектури

Головними перевагами такої архітектури є:

- Простота системи, у порівняні з тришаровою і багатошаровою архітектурами;

- Гарантія цілісності даних;

- Повна підтримка одночасної роботи багатьох користувачів.

Але така архітектура має досить значні недоліки, а саме:

- Необхідність більш потужного комп’ютера в якості сервера та потужних клієнтських машин, здатних забезпечити і бізнес логіку і графічний інтерфейс;

- Відсутність масштабування. Слабкий захист від взлому;

- Бізнес-логіка повністю на стороні клієнта. При її зміні треба повністю оновлювати клієнтське ПЗ.

Через недоліки двошарової архітектури на зміну їй прийшла трьохшарова

На рисунку 1.3 представлено приклад трьохшарової архітектури



Рисунок 1.3 - Приклад трьохшарової архітектури

Основними шарами даної архітектури є шар представлення; домен, котрий ще називають шаром бізнес-логіки та шар даних, який узагальнює джерела даних. Кожен з них має визначені функції та несе відповідальність за частину роботи, виконувану додатком, шари можуть розміщуватися не тільки локально на одному пристрої, а і бути розділеними, наприклад представлення на клієнтській частині, а бізнес-логіка і джерело даних – на серверній частині додатку

Шар представлення виконує надання послуг, відображення даних, обробку подій користувацького інтерфейсу, обслуговування HTTP-запитів, підтримку функцій командної строки та API пакетного використання.

Шар домену - бізнес-логіку додатку, специфічні алгоритми

Джерело даних - запити до бази даних, обмін повідомленнями, управління транзакціями, тощо

Шару представлення стосується усе, що пов’язане зі взаємодією користувача з системою. Він може бути простим, як командна строка чи текстове меню, але зараз користувачу, ймовірніше за все, доведеться мати справу з графічним інтерфейсом, оформленим у стилі «товстого» клієнта.

Головна задача шару представлення – транслювати команди користувача у формат, зрозумілий шару бізнес-логіки.

Логіка домену – описує основні функції додатку, призначені для досягнення поставленої перед ним цілі. До цих функцій належать обчислення на основі введених і збережених процедур, перевірка усіх елементів даних і обробка команд, що надходять від шару представлення, а також передача інформації шару джерела даних.

Іноді шари організовують таким чином, щоб бізнес-логіка повністю приховувала джерело даних від представлення. Однак частіше код представлення може звертатися до джерела даних безпосередньо. Хоча такий варіант менш бездоганний з теоретичної точки зору, в практичному використанні він нерідко більш зручний та доцільний; код представлення може інтерпретувати команду користувача, активізувати функції джерела даних для отримання відповідних порцій інформації з бази даних, звернутися до засобів бізнес-логіки для аналізу цієї інформації і виконання необхідних розрахунків і тільки після цього відобразити відповідну картину на екрані.

Джерело даних – це підмножина функцій, що забезпечують взаємодію зі сторонніми системами, котрі виконують завдання в інтересах додатку. Код цієї категорії несе відповідальність за моніторинг транзакцій, управління іншими додатками, обмін повідомленнями тощо. Для більшості корпоративних додатків основна частина логіки джерела даних концентрується в коді СКБД

Індустрія не стояла на місці та розширила поняття трирівневої архітектури до багаторівневої. Логічно модель має таку ж саму структуру, але всеохоплююче використання Інтернету внесло свої корективи, ставши важливою частиною багатьох програмних додатків.

Веб-сервіси (а пізніше REST дані) стали більш інтегровані в додатки. Як наслідок, шар даних, як правило, стали розщеплювати на рівень зберігання даних (сервер баз даних) і рівень доступу до даних. У комплексних системах для уніфікації доступу до баз даних і веб-сервісів розробляють додатковий рівень класів-обгорток.

Веб-браузери були менш потужним, ніж традиційні додатки клієнтського рівня і логіка користувацького інтерфейсу розділилися між браузером з JavaScript і сервером з додатком веб-сервера, що містить у собі логіку користувацького інтерфейсу.

Шари все далі і далі набували більш розмитого характеру із додаванням збережуваних процедур усіма основними постачальниками баз даних і баз даних з відкритим вихідним кодом. Це призвело до поширення практики переносу деяких частин бізнес-логіки від бізнес-рівня на рівень бази даних, тобто з’явилася концепція створення рівнів в межах рівнів.

Так як під впливом Інтернету, технологічних інновацій і сервісів архітектура додатку стала більш розмитою, трьохшарова модель додатку розвинулася у багаторівневу архітектуру

На рисунку 1.4 представлено приклад багатошарової архітектури



Рисунок 1.4 – Приклад багатошарової архітектури

Багатошарова архітектура з’явилася завдяки створенню декількох рівнів в інших рівнях, а саме в рівні представлення:

- компоненти графічного інтерфейсу, котрі відповідають за відображення графічних елементів;

- компоненти процесів графічного інтерфейсу, котрі реагують на події, що відбуваються у графічному інтерфейсі.

Великі корпоративні додатки часто структуровані навколо бізнеспроцесів та бізнес-компонентів. Ці поняття розглядаються в рамках цілого ряду компонентів, сутностей, агентів та інтерфейсів бізнес-рівня:

- бізнес-компоненти – програмні реалізації концепцій чи процесів. Вони складаються з усіх артефактів необхідних для представлення, реалізації, розгортування конкретної концепції як автономного елемента більшої системи, котрий можна використовувати повторно;

- бізнес-сутності – це структури, що виступають контейнерами даних. Вони інкапсулюють та приховують деталі специфічного формату представлення даних. Наприклад, бізнес сутність може інкапсулювати набір записів, отриманих з бази даних. Пізніше, ця ж бізнес-сутність може бути змінена для огортання в XML-документ з мінімальним впливом на інші частини додатку;

- сервісні інтерфейси – додаток може надавати частину його функціоналу як сервіс, котрий можуть використовувати інші додатки. В ідеалі він приховує деталі реалізації і надає тільки тонкий шар інтерфейсу;

- бізнес-процеси – відображають діяльність бізнесу на високих рівні абстракції системи, як-то обробка замовлення, підтримка користувача, закупка товару.

Шар даних теж зазнав певних внутрішніх метаморфоз, внаслідок чого з’явилися наступні шари:

- компоненти доступу до даних – ізолюють бізнес-шар від деталей реалізації, специфічних для сховища даних. Дозволяє мінімізувати вплив зміни постачальника бази даних, зміни представлення даних, наприклад, схеми бази даних, інкапсулює весь код, що маніпулює конкретною одиницею даних в одному місці, що надзвичайно спрощує підтримку та тестування;

- сервісні шлюзи – бізнес-компоненти часто повинні отримувати доступ до внутрішніх та зовнішніх сервісів чи додатків. Сервісний шлюз – це компонент, що інкапсулює інтерфейс, протокол та код, потрібний для використання сервісів. Наприклад, бізнес-рішення часто потребує інформацію з деякого сервісу для завершення бізнес-процесу. Воно делегуватиме всю взаємодію з цим сервісом шлюзу. Сервісний шлюз надає можливість з меншими зусиллями змінити зовнішній сервіс на інший. Також даний підхід надає змогу емулювати зовнішній сервіс, наприклад, для тестування доменного рівня.

Надодачу до описаних шарів багатошарова архітектура визначає набір фундаментальних сервісів, котрі потенційно можуть використовувати усі інші шари. Ці сервіси діляться на три базові категорії:

- шар безпеки – сервіси цього шару підтримують безпеку додатку;

- шар операційного управління – ці сервіси оперують компонентами і зв’язаними з ними ресурсами і також торкаються таких вимог як масштабованість та відмовостійкість;

- шар сервісів комунікації – сервіси, котрі надають можливість спілкуватися різним шарам між собою.

Переваги даної архітектури – гарна точка відправлення для побудови власних додатків. Розробнику, що використовує даний підхід, дістаються найбільші позитивні риси розшарованого додатку. Але є й певні аспекти архітектури, які додають відповідальності, а саме, для важких, комплексних рішень необхідно правильно розділяти доменний рівень, особливо, якщо можливість повторного використання компонентів є в пріоритеті або якщо розробник проектує сімейство рішень, що базується на наборі компонентів. У такому випадку типовим є заміна одного бізнес-шару класичного тришарового додатку трьома.

**3 СТВОРЕННЯ ПОСТАНОВИ ЗАВДАННЯ ДО ДИПЛОМНОЇ РОБОТИ**

Метою курсової роботи є створення системи онлайн продаж. Дана система здійснює продаж аксесуарів для домашніх тварин.

Система повинна дозволити покупцям здійснювати швидкий та детальний перегляд продукції.

Сервіс повинен включати в себе функцію реєстрації користувача, залишення відгуків та здійснення замовлення, додавання товару.

Після вивчення предметної області і створення відповідної бази даних необхідно створити сервіс, який має наступні функції:

- Вибір необхідних товарів с фільтрацією по будь-якій характеристиці;

- Створення аккаунтів, зміни інформації про них

- Додавання та видалення відгуків про товар;

- Створення замовлень, перегляд інформації про них;

- Зручне та швидке маніпулювання адміністратором даними про товари(додавання, видалення, змінення).

Повинна бути реалізована 2-х рівнева система адміністрації: модератори та адміністратори

Вимоги до надійності та безпеки:

- Тільки адміністратор має право додавання, змінення чи видалення товару;

- Модератор має право видаляти відгуки про товари та змінювати статуси замовлення;

- Адміністратор має право змінювати інформацію про постачальників, категорії та типи товарів.

Етапами розробки програмного проєкту є:

- Формування вимог;

- Проєктування системи;

- Реалізація;

- Тестування та виправлення помилок, якщо такі є;

- Впровадження;

- Експлуатація та супровід.

**4 ПРОГРАМУВАННЯ**

**4.1 Проектування інтерфейсу користувача**

Основним призначення програми є надання клієнтській частині потрібну їй інформацію за допомогою запитів.

Оскільки вся програма розбита на модулі внесення подальших змін буде максимально швидким та мати мінімальну кількість перешкод

Основними модулями програми є

- Моделі - схеми та моделі mongoose, які описують дані MongoDB;

- Роути – уловлювачі запитів клієнтської частини, які потім викликають потрібні контролери;

- Контролери – Головні обробники програми, які за допомогою mongoose витягують дані з MongoDB, обробляють дані враховуючи параметри запиту та повертають результат;

- Хелпери – Допоміжні функції програми;

- Сваггер – Опис роутів для створення візульного інтерфейсу.

Мінімальні вимоги до середовища функціонування системи:

- CPU: 1 vCPU Type-A;

- RAM: 1 Гибибайт;

- SSD: 2 Гибибайт;

- Інтернет трафік на місяць: 3 Тибибайт;

- Операційна система: Ubuntu, Debian, Arch;

- Встановлена база даних: MongoDB 4.4;

- Програмне оточення: NodeJS.

**4.2 Створення головного модуля програми**

Перед запуском сервісу треба створити змінні оточення:

- Змінна PORT – Відповідає за порт на якому буде працювати сервіс;

- Змінна MONGO\_USERNAME – Відповідає за ім’я користувача, яке буде використане при авторизації в MongoDB ;

- Змінна MONGO\_PASSWORD – Відповідає за пароль користувача, який буде використаний при авторизації в MongoDB;

- Змінна MONGO\_DATABASE – Відповідає за ім’я бази даних.

Для запуску сервісу треба виконати команду: node index.js

Після запуску, в браузері потрібно відкрити сторінку сервісу за посиланням <http://localhost:8000/api-docs>

Вигляд сторінки сервісу продемонстрований на рисунку 2.30

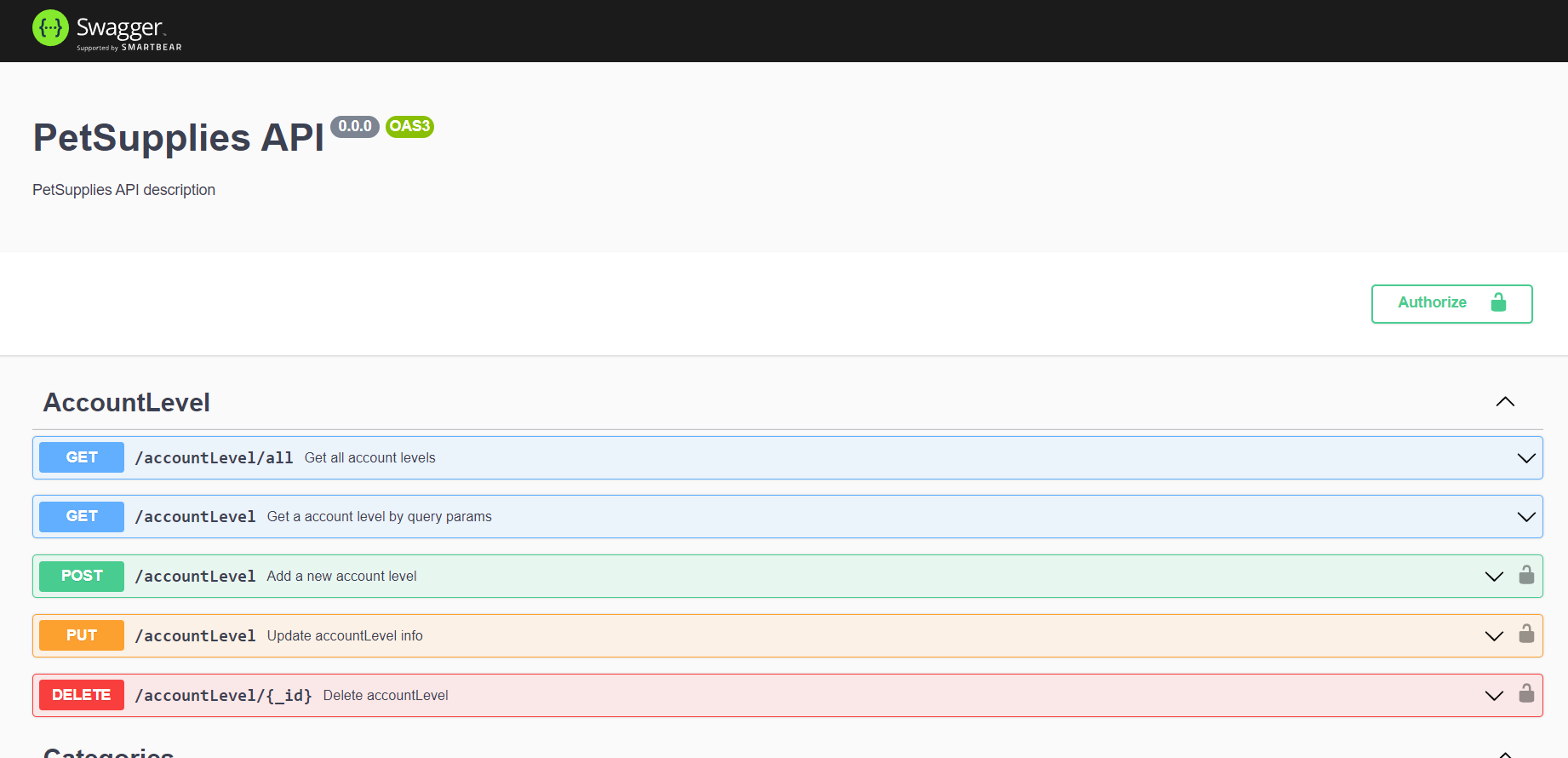


Рисунок 2.30 – Вигляд сторінки сервісу API

Всі можливості сервісу поділені на категорії, які можна згортати при необхідності  
На прикладі категорії AccountLevel розглянемо основні можливості сервісу. Як видно з рисунку 2.30 сервіс надає можливість робити операції з даними:

- взяття(GET);

- створення або додавання(POST);

- зміна(PUT);

- видалення(DELETE)

На операціях додавання, змінювання та видалення на рисунку 2.30 є замок, який говорить про те, що для виконання цієї операції користувач повинен бути авторизований в системі та мати необхідний рівень доступу

На рисунку 2.30 всі операції знаходяться в згорнутому вигляді

Для виконання операції потрібно розгорнути вкладку з нею, після чого на екрані користувача з’явиться схема запросу, яка включає в себе параметри або ж тіло запиту та відповіді(Responses) серверу.

Приклад тіла запиту представлений на рисунку 2.31

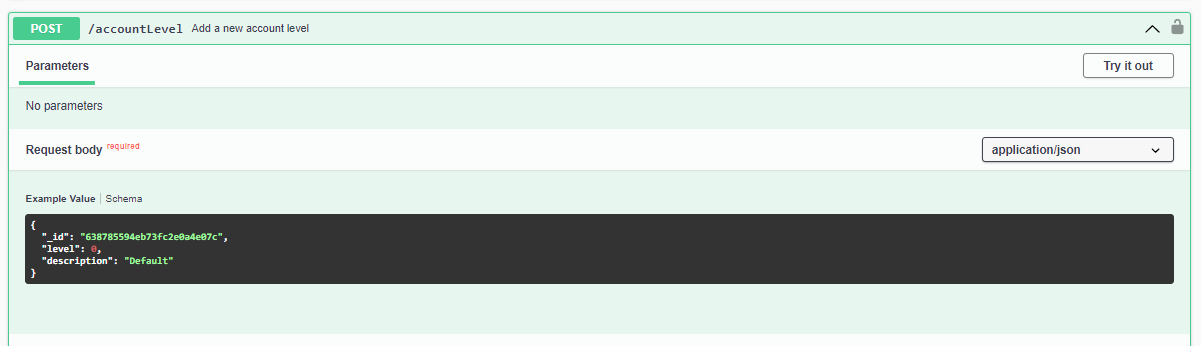


Рисунок 2.31 – Приклад тіла запиту

В відповідях запиту можливо переглянути що саме поверне сервер користувачу після виконання запиту.

При успішному запиті сервер поверне відповідь з кодом 200 та відповідний до прикладу результат.

При помилкі на сервері, або в параметрах запиту повернеться 500 помилка з відповідним до прикладу результатом

При помилці авторизації сервер поверне 401 помилку та повідомлення

Приклад відповідей серверу представлені на рисунку 2.32

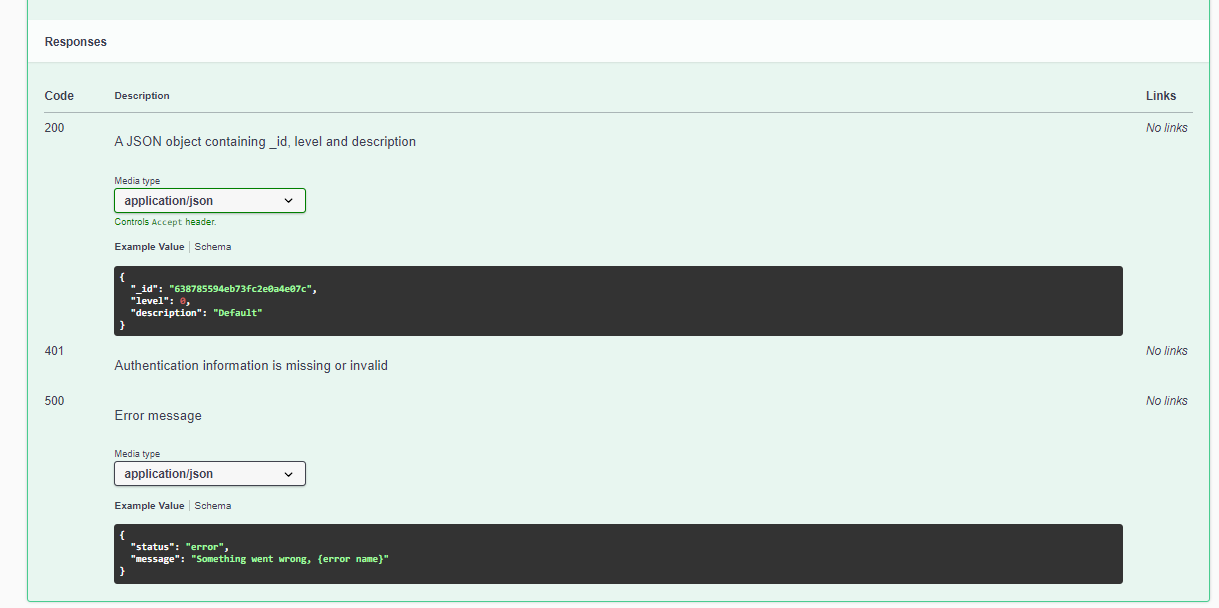


Рисунок 2.32 – Приклад відповідей запиту

Для виконання запиту потрібно натиснути на кнопку «Try it out», ввести необхідні дані та натиснути кнопку «Execute»

Після виконання запиту в секції відповідей(Responses) з’явиться curl(команда, яку виконала клієнтська частина), Request url (посилання, на яке був виконаний запит), а нижче код, тіло та заголовки відповіді серверу

Приклад виконаного запиту представлений на рисунку 2.33

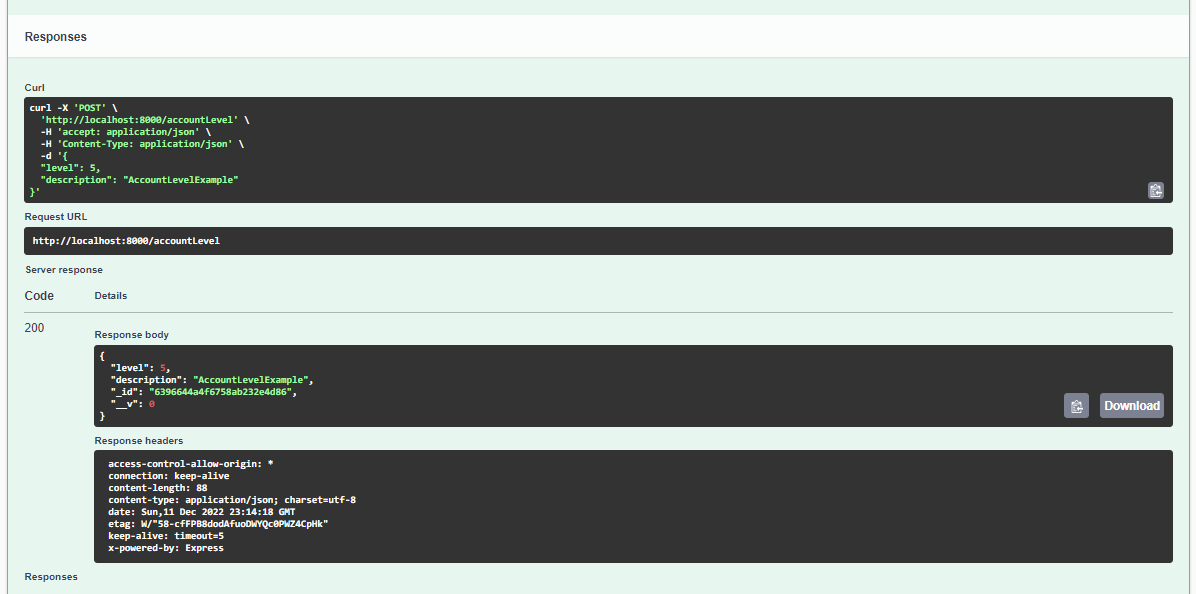


Рисунок 2.33 – Приклад виконаного запиту

**ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ**

1 http://deep.kiev.ua/attachments/093\_2016d\_Kravchuk.pdf

2 <https://code.visualstudio.com/#meet-intellisense>

3 <https://git-scm.com/>

4 <https://www.postman.com/>

5 <https://swagger.io/>

6 https://developer.mozilla.org/ru/docs/conflicting/Web/JavaScript

7 <https://www.typescriptlang.org/>

8 https://nodejs.org/en/about/

9 https://github.com/nvm-sh/nvm

10 https://www.npmjs.com/about

11 https://pnpm.io/motivation

12 https://aws.amazon.com/ru/documentdb/what-is-mongodb

13 <https://www.freecodecamp.org/news/introduction-to-mongoose-> for-mongodb-d2a7aa593c57

14 https://developer.mozilla.org/ru/docs/Web/HTTP

15 <https://www.ukraine.com.ua/blog/seo-optimization/chto-takoe->https-i-zachem-on-nuzhen-kazhdomu-sajtu.html

16 https://habr.com/ru/post/38730/

17 https://expressjs.com/

18 https://swagger.io/docs/specification/about/

19 https://prettier.io/docs/en/index.html

20 https://dev.to/shivambmgupta/eslint-what-why-when-how-5f1d

21 <https://www.npmjs.com/package/nodemon>

22 https://www.postman.com/

23 І. О. Бардус, М. І. Лазарєв, А. О. Ніценко БАЗИ ДАНИХ У СХЕМАХ (НА ОСНОВІ ФУНДАМЕНТАЛІЗОВАНОГО ПІДХОДУ)]

24 <https://www.geeksforgeeks.org/what-is-mongodb-working-and->features/

**ДОДАТОК А**

**Текст головного модуля програми**

src\index.ts

import dotenv from'dotenv';import express from'express';import{connect,mongo}from'mongoose';import cors from'cors';import swaggerUI from'swagger-ui-express';import swaggerJsDoc from'swagger-jsdoc';import{HomeRouter,AccountLevelRouter,CategoriesRouter,ProducerRouter,TypesRouter,OrdersRouter,}from'./routes';import mongoose from'mongoose';import bodyParser from'body-parser';console.log(new mongoose.Types.ObjectId('638387eb6f33fac689be5e82'));dotenv.config();connect(`mongodb+srv:).then(()=>{console.log('[Server]: DB connected')}).catch(err=>{console.error('[Server]: DB error connect',err)});const app=express();const port=process.env.PORT;app.use(cors());const options={failOnErrors:true,definition:{openapi:'3.0.0',info:{title:'PetSupplies API',version:'0.0.0',description:'PetSupplies API description',servers:[`http:},},apis:['src/routes/\*.ts','src/swagger/\*.yaml'],};const openapiSpecification=swaggerJsDoc(options);app.use('/api-docs',swaggerUI.serve,swaggerUI.setup(openapiSpecification));app.use(bodyParser.json());app.use(bodyParser.urlencoded({extended:true}));app.use(HomeRouter);app.use(CategoriesRouter);app.use(TypesRouter);app.use(ProducerRouter);app.use(AccountLevelRouter);app.use(OrdersRouter);app.listen(port,()=>{console.log(`[Server]:Server is running at http:})

src\controllers\AccountLevel\createNewAccountLevel.ts

import mongoose,{Types}from'mongoose';import{Request,Response}from'express';import{AccountLevel}from'../../models';import{Status,ErrorMessageAnswer,Codes}from'../../types';import{createErrorMessage}from'../../helpers/messages';import{getValidParamsWithCheckID}from'../../helpers/getValidParams';type getQueryParamsOk={status:Status.OK;value:{id:Types.ObjectId|undefined;level:number;description:string}};const getQueryParams=(body:Request['body']):ErrorMessageAnswer|getQueryParamsOk=>{const{\_id,level,description}=body;let resultId:undefined|Types.ObjectId=undefined;if(\_id!==undefined){if(typeof \_id!=='string'){return createErrorMessage('Id must be undefined | string')};if(!Types.ObjectId.isValid(\_id)){return createErrorMessage('Id string must be ObjectId isValid')};resultId=new Types.ObjectId(\_id)};if(typeof description!=='string'){return createErrorMessage('Description must be string')};let resultLevel:number|undefined=undefined;resultLevel=Number(level);if(resultLevel===undefined||resultLevel===NaN){return createErrorMessage('Level must be number | undefined')};return{status:Status.OK,value:{id:resultId,level:resultLevel,description},}};export const createNewAccountLevel=async(req:Request,res:Response)=>{const params=getQueryParams(req.body);if(params.status===Status.ERROR){return res.status(Codes.ERROR).json(params)};const newAccountLevelParams=getValidParamsWithCheckID(params.value);new AccountLevel(newAccountLevelParams).save((error,result)=>{if(error){return res.status(Codes.ERROR).json(createErrorMessage('Something went wrong.'+error.name))};return res.status(Codes.OK).json(result)})}

src\controllers\AccountLevel\deleteAccountLevel.ts

import{Types}from'mongoose';import{Request,Response}from'express';import{AccountLevel}from'../../models';import{Codes}from'../../types';import{createErrorMessage,createSuccessMessage,}from'../../helpers/messages';export const deleteAccountLevel=(req:Request,res:Response)=>{const id=req.params.\_id;if(typeof id!=='string'){return createErrorMessage('Id must be string')};if(!Types.ObjectId.isValid(id)){return createErrorMessage('Id string must be ObjectId.isValid')};const idAsObjectId=new Types.ObjectId(id);AccountLevel.deleteOne({\_id:idAsObjectId},error=>{if(error){return res.status(Codes.ERROR).json(createErrorMessage('Something went wrong.'+error.name))};return res.status(Codes.OK).json(createSuccessMessage(`AccountLevel ${id}successfully deleted`))})}

src\controllers\AccountLevel\getAccountLevel.ts

import{Types}from'mongoose';import{Request,Response}from'express';import{AccountLevel}from'../../models';import{Status,ErrorMessageAnswer,Codes}from'../../types';import{createErrorMessage}from'../../helpers/messages';import{getValidParamsWithCheckAll}from'../../helpers/getValidParams';type getQueryParamsOk={status:Status.OK;value:{id:Types.ObjectId|undefined;level:number|undefined;description:string|undefined}};const getQueryParams=(req:Request):ErrorMessageAnswer|getQueryParamsOk=>{const{id,level,description}=req.query;let resultId:undefined|Types.ObjectId=undefined;if(id!==undefined){if(typeof id!=='string'){return createErrorMessage('Id must be undefined | string')};if(!Types.ObjectId.isValid(id)){return createErrorMessage('Id string must be ObjectId.isValid')};resultId=new Types.ObjectId(id)};if(description!==undefined){if(typeof description!=='string'){return createErrorMessage('Description must be string')}};let resultLevel:number|undefined=undefined;if(level!==undefined){if(Number(level)===NaN){return createErrorMessage('Level must be number | undefined')};resultLevel=Number(level)};return{status:Status.OK,value:{id:resultId,level:resultLevel,description},}};export const getAccountLevel=async(req:Request,res:Response)=>{const params=getQueryParams(req);if(params.status===Status.ERROR){return res.status(Codes.ERROR).json(params)};const findParams=getValidParamsWithCheckAll(params.value);AccountLevel.findOne(findParams,{\_\_v:0}).exec((error,result)=>{if(error){return res.status(Codes.ERROR).json(createErrorMessage('Something went wrong.'+error.name))};return res.status(Codes.OK).json(result)})}

src\controllers\AccountLevel\getAllAccountLevels.ts

import{Request,Response}from'express';import{AccountLevel}from'../../models';import{Codes}from'../../types';import{createErrorMessage}from'../../helpers/messages';export const getAllAccountLevels=(req:Request,res:Response)=>{AccountLevel.find({},{\_\_v:0}).exec((error,result)=>{if(error){return res.status(Codes.ERROR).json(createErrorMessage('Something went wrong.'+error.name))};return res.status(Codes.OK).json(result)})}

src\controllers\AccountLevel\index.ts

export{createNewAccountLevel}from'./createNewAccountLevel';export{deleteAccountLevel}from'./deleteAccountLevel';export{getAccountLevel}from'./getAccountLevel';export{updateAccountLevel}from'./updateAccountLevel';export{getAllAccountLevels}from''

src\controllers\AccountLevel\updateAccountLevel.ts

import{Types}from'mongoose';import{Request,Response}from'express';import{AccountLevel}from'../../models';import{Status,ErrorMessageAnswer,Codes}from'../../types';import{createErrorMessage}from'../../helpers/messages';import{getValidParamsWithCheckAll}from'../../helpers/getValidParams';type getQueryParamsOk={status:Status.OK;value:{id:Types.ObjectId;level:number|undefined;description:string|undefined}};const getQueryParams=(req:Request):ErrorMessageAnswer|getQueryParamsOk=>{const{id,level,description}=req.query;if(typeof id!=='string'){return createErrorMessage('Id must be undefined | string')};if(!Types.ObjectId.isValid(id)){return createErrorMessage('Id string must be ObjectId.isValid')};const resultId=new Types.ObjectId(id);if(description!==undefined){if(typeof description!=='string'){return createErrorMessage('Description must be string')}};let resultLevel:number|undefined=undefined;if(level!==undefined){if(Number(level)===NaN){return createErrorMessage('Level must be number | undefined')};resultLevel=Number(level)};return{status:Status.OK,value:{id:resultId,level:resultLevel,description},}};export const updateAccountLevel=async(req:Request,res:Response)=>{const params=getQueryParams(req);if(params.status===Status.ERROR){return res.status(Codes.ERROR).json(params)};const{id,level,description}=params.value;const updatedParams=getValidParamsWithCheckAll({id,level,description,});AccountLevel.updateOne({\_id:id},updatedParams,null,error=>{if(error){return res.status(Codes.ERROR).json(createErrorMessage('Something went wrong.'+error.name))};AccountLevel.findOne({\_id:id}).exec((findError,findResult)=>{if(findError){return res.status(Codes.ERROR).json(createErrorMessage('Something went wrong.'+findError.name))};return res.status(Codes.OK).json(findResult)})})}

src\controllers\Category\createNewCategory.ts

import mongoose,{Types}from'mongoose';import{Request,Response}from'express';import{Category,User}from'../../models';import{Status,SuccessMessageAnswer,ErrorMessageAnswer,Codes,}from'../../types';import{createErrorMessage,createSuccessMessage,}from'../../helpers/messages';import{getValidParamsWithCheckID}from'../../helpers/getValidParams';const Buffer=mongoose.Types.Buffer;type getQueryParamsOk={status:Status.OK;value:{id:Types.ObjectId|undefined;title:string;description:string}};const getQueryParams=(body:Request['body']):ErrorMessageAnswer|getQueryParamsOk=>{const{\_id,title,description}=body;let resultId:undefined|Types.ObjectId=undefined;if(\_id!==undefined){if(typeof \_id!=='string'){return createErrorMessage('Id must be undefined | string')};if(!Types.ObjectId.isValid(\_id)){return createErrorMessage('Id string must be ObjectId isValid')};resultId=new Types.ObjectId(\_id)};if(typeof title!=='string'){return createErrorMessage('Title must be string')};if(typeof description!=='string'){return createErrorMessage('Description must be string')};return{status:Status.OK,value:{id:resultId,title,description},}};export const createNewCategory=async(req:Request,res:Response)=>{const params=getQueryParams(req.body);if(params.status===Status.ERROR){return res.status(Codes.ERROR).json(params)};const{id,title,description}=params.value;const newCategoryParams=getValidParamsWithCheckID({id,title,description,});new Category(newCategoryParams).save((error,result)=>{if(error){return res.status(Codes.ERROR).json(createErrorMessage('Something went wrong.'+error.name))};return res.status(Codes.OK).json(result)})}

src\controllers\Category\deleteCategory.ts

import{Types}from'mongoose';import{Request,Response}from'express';import{Category,User}from'../../models';import{Codes}from'../../types';import{createErrorMessage,createSuccessMessage,}from'../../helpers/messages';export const deleteCategory=(req:Request,res:Response)=>{const id=req.params.\_id;if(typeof id!=='string'){return createErrorMessage('Id must be string')};if(!Types.ObjectId.isValid(id)){return createErrorMessage('Id string must be ObjectId.isValid')};const idAsObjectId=new Types.ObjectId(id);Category.deleteOne({\_id:idAsObjectId},error=>{if(error){return res.status(Codes.ERROR).json(createErrorMessage('Something went wrong.'+error.name))};return res.status(Codes.OK).json(createSuccessMessage(`Category ${id}successfully deleted`))})}

src\controllers\Category\getAllCategories.ts

import{Request,Response}from'express';import{Category}from'../../models';import{Codes}from'../../types';import{createErrorMessage}from'../../helpers/messages';export const getAllCategories=(req:Request,res:Response)=>{Category.find({},{\_\_v:0}).exec((error,result)=>{if(error){return res.status(Codes.ERROR).json(createErrorMessage('Something went wrong.'+error.name))};return res.status(Codes.OK).json(result)})}

src\controllers\Category\getCategories.ts

import{Types}from'mongoose';import{Request,Response}from'express';import{Category,User}from'../../models';import{Status,ErrorMessageAnswer,Codes}from'../../types';import{createErrorMessage}from'../../helpers/messages';import{getValidParamsWithCheckAll}from'../../helpers/getValidParams';type getQueryParamsOk={status:Status.OK;value:{id:Types.ObjectId|undefined;title:string|undefined;description:string|undefined}};const getQueryParams=(req:Request):ErrorMessageAnswer|getQueryParamsOk=>{const{id,title,description}=req.query;let resultId:undefined|Types.ObjectId=undefined;if(id!==undefined){if(typeof id!=='string'){return createErrorMessage('Id must be undefined | string')};if(!Types.ObjectId.isValid(id)){return createErrorMessage('Id string must be ObjectId.isValid')};resultId=new Types.ObjectId(id)};if(title!==undefined){if(typeof title!=='string'){return createErrorMessage('Title must be string')}};if(description!==undefined){if(typeof description!=='string'){return createErrorMessage('Description must be string')}};return{status:Status.OK,value:{id:resultId,title,description},}};export const getCategories=async(req:Request,res:Response)=>{const params=getQueryParams(req);if(params.status===Status.ERROR){return res.status(Codes.ERROR).json(params)};const{id,title,description}=params.value;const findParams=getValidParamsWithCheckAll({id,title,description});Category.find(findParams,{\_\_v:0}).exec((error,result)=>{if(error){return res.status(Codes.ERROR).json(createErrorMessage('Something went wrong.'+error.name))};return res.status(Codes.OK).json(result)})}

src\controllers\Category\index.ts

export{createNewCategory}from'./createNewCategory';export{getCategories}from'./getCategories';export{deleteCategory}from'./deleteCategory';export{updateCategory}from'./updateCategory';export{getAllCategories}from''

src\controllers\Category\updateCategory.ts

import{Types}from'mongoose';import{Request,Response}from'express';import{Category,User}from'../../models';import{Status,ErrorMessageAnswer,Codes}from'../../types';import{createErrorMessage}from'../../helpers/messages';import{getValidParamsWithCheckAll}from'../../helpers/getValidParams';type getQueryParamsOk={status:Status.OK;value:{id:Types.ObjectId;title:string|undefined;description:string|undefined}};const getQueryParams=(req:Request):ErrorMessageAnswer|getQueryParamsOk=>{const{id,title,description}=req.query;if(typeof id!=='string'){return createErrorMessage('Id must be string')};if(!Types.ObjectId.isValid(id)){return createErrorMessage('Id string must be ObjectId.isValid')};const resultId=new Types.ObjectId(id);if(title!==undefined){if(typeof title!=='string'){return createErrorMessage('Title must be string')}};if(description!==undefined){if(typeof description!=='string'){return createErrorMessage('Description must be string')}};return{status:Status.OK,value:{id:resultId,title,description},}};export const updateCategory=async(req:Request,res:Response)=>{const params=getQueryParams(req);if(params.status===Status.ERROR){return res.status(Codes.ERROR).json(params)};const{id,title,description}=params.value;const updatedParams=getValidParamsWithCheckAll({id,title,description});Category.updateOne({\_id:id},updatedParams,null,(error,result)=>{if(error){return res.status(Codes.ERROR).json(createErrorMessage('Something went wrong.'+error.name))};Category.findOne({\_id:id}).exec((findError,findResult)=>{if(findError){return res.status(Codes.ERROR).json(createErrorMessage('Something went wrong.'+findError.name))};return res.status(Codes.OK).json(findResult)})})}

src\controllers\Order\createNewOrder.ts

import mongoose,{Types}from'mongoose';import{Request,Response}from'express';import{Order,Product,User}from'../../models';import{Status,SuccessMessageAnswer,ErrorMessageAnswer,Codes,}from'../../types';import{createErrorMessage,createSuccessMessage,}from'../../helpers/messages';import{getValidParamsWithCheckID}from'../../helpers/getValidParams';type getQueryParamsOk={status:Status.OK;value:{id:Types.ObjectId|undefined;user:Types.ObjectId;product:Types.ObjectId;count:number;productPrice:number|undefined;address:string|undefined;status:string|undefined}};const getQueryParams=(body:Request['body']):ErrorMessageAnswer|getQueryParamsOk=>{const{id,user,product,count,productPrice,address,status}=body;let resultId:undefined|Types.ObjectId=undefined;let resultUser:undefined|Types.ObjectId=undefined;let resultProduct:undefined|Types.ObjectId=undefined;if(id!==undefined){if(typeof id!=='string'){return createErrorMessage('\_id must be string')};if(!Types.ObjectId.isValid(id)){return createErrorMessage('\_id string must be ObjectId.isValid')};resultId=new Types.ObjectId(id)};if(typeof user!=='string'){return createErrorMessage('user must be undefined | string')};if(!Types.ObjectId.isValid(user)){return createErrorMessage('user string must be ObjectId.isValid')};resultUser=new Types.ObjectId(user);if(typeof product!=='string'){return createErrorMessage('product must be undefined | string')};if(!Types.ObjectId.isValid(product)){return createErrorMessage('product string must be ObjectId.isValid')};resultProduct=new Types.ObjectId(product);let resultCount:undefined|number=undefined;let resultProductPrice:undefined|number=undefined;if(Number(count)===NaN){return createErrorMessage('address must be string')};resultCount=Number(count);if(productPrice!==undefined){if(Number(productPrice)===NaN){return createErrorMessage('address must be string')};resultProductPrice=Number(productPrice)};if(address!==undefined){if(typeof address!=='string'){return createErrorMessage('address must be string')}};if(status!==undefined){if(typeof status!=='string'){return createErrorMessage('status must be string')}};return{status:Status.OK,value:{id:resultId,user:resultUser,product:resultProduct,count:resultCount,productPrice:resultProductPrice,address,status,},}};const checkProductIsExists=async(id:Types.ObjectId):Promise<boolean>=>{const productResult=await Product.findOne({\_id:id});return productResult!==null};const checkUserIsExists=async(id:Types.ObjectId):Promise<boolean>=>{const result=await User.findOne({\_id:id});return result!==null};export const createNewOrder=async(req:Request,res:Response)=>{const params=getQueryParams(req.body);if(params.status===Status.ERROR){return res.status(Codes.ERROR).json(params)};const newOrderParams=getValidParamsWithCheckID<getQueryParamsOk['value']>(params.value);if((await checkProductIsExists(newOrderParams.product))===false){return res.status(Codes.ERROR).json(createErrorMessage('Product id is not valid'))};if((await checkUserIsExists(newOrderParams.user))===false){return res.status(Codes.ERROR).json(createErrorMessage('User id is not valid'))};new Order(newOrderParams).save((error,result)=>{if(error){return res.status(Codes.ERROR).json(createErrorMessage('Something went wrong.'+error.name))};return res.status(Codes.OK).json(result)})}

src\controllers\Order\deleteOrder.ts

import{Types}from'mongoose';import{Request,Response}from'express';import{Order}from'../../models';import{Codes}from'../../types';import{createErrorMessage,createSuccessMessage,}from'../../helpers/messages';export const deleteOrder=(req:Request,res:Response)=>{const id=req.params.\_id;if(typeof id!=='string'){return createErrorMessage('Id must be string')};if(!Types.ObjectId.isValid(id)){return createErrorMessage('Id string must be ObjectId.isValid')};const idAsObjectId=new Types.ObjectId(id);Order.deleteOne({\_id:idAsObjectId},error=>{if(error){return res.status(Codes.ERROR).json(createErrorMessage('Something went wrong.'+error.name))};return res.status(Codes.OK).json(createSuccessMessage(`Order ${id}successfully deleted`))})}

src\controllers\Order\getAllOrders.ts

import{Request,Response}from'express';import{Order}from'../../models';import{Codes}from'../../types';import{createErrorMessage}from'../../helpers/messages';export const getAllOrders=(req:Request,res:Response)=>{Order.find({},{\_\_v:0}).exec((error,result)=>{if(error){return res.status(Codes.ERROR).json(createErrorMessage('Something went wrong.'+error.name))};return res.status(Codes.OK).json(result)})}

src\controllers\Order\getOrder.ts

import{Types}from'mongoose';import{Request,Response}from'express';import{Order}from'../../models';import{Status,ErrorMessageAnswer,Codes}from'../../types';import{createErrorMessage}from'../../helpers/messages';import{getValidParamsWithCheckAll}from'../../helpers/getValidParams';type getQueryParamsOk={status:Status.OK;value:{id:Types.ObjectId|undefined;user:Types.ObjectId|undefined;product:Types.ObjectId|undefined;count:number|undefined;productPrice:number|undefined;address:string|undefined;status:string|undefined}};const getQueryParams=(req:Request):ErrorMessageAnswer|getQueryParamsOk=>{const{id,user,product,count,productPrice,address,status}=req.query;let resultId:undefined|Types.ObjectId=undefined;let resultUser:undefined|Types.ObjectId=undefined;let resultProduct:undefined|Types.ObjectId=undefined;if(id!==undefined){if(typeof id!=='string'){return createErrorMessage('Id must be undefined | string')};if(!Types.ObjectId.isValid(id)){return createErrorMessage('Id string must be ObjectId.isValid')};resultId=new Types.ObjectId(id)};if(user!==undefined){if(typeof user!=='string'){return createErrorMessage('user must be undefined | string')};if(!Types.ObjectId.isValid(user)){return createErrorMessage('user string must be ObjectId.isValid')};resultUser=new Types.ObjectId(user)};if(product!==undefined){if(typeof product!=='string'){return createErrorMessage('product must be undefined | string')};if(!Types.ObjectId.isValid(product)){return createErrorMessage('product string must be ObjectId.isValid')};resultProduct=new Types.ObjectId(product)};let resultCount:undefined|number=undefined;let resultProductPrice:undefined|number=undefined;if(count!==undefined){if(Number(count)===NaN){return createErrorMessage('address must be string')};resultCount=Number(count)};if(productPrice!==undefined){if(Number(productPrice)===NaN){return createErrorMessage('address must be string')};resultProductPrice=Number(productPrice)};if(address!==undefined){if(typeof address!=='string'){return createErrorMessage('address must be string')}};if(status!==undefined){if(typeof status!=='string'){return createErrorMessage('status must be string')}};return{status:Status.OK,value:{id:resultId,user:resultUser,product:resultProduct,count:resultCount,productPrice:resultProductPrice,address,status,},}};export const getOrder=async(req:Request,res:Response)=>{const params=getQueryParams(req);if(params.status===Status.ERROR){return res.status(Codes.ERROR).json(params)};const findParams=getValidParamsWithCheckAll(params.value);Order.find(findParams,{\_\_v:0}).exec((error,result)=>{if(error){return res.status(Codes.ERROR).json(createErrorMessage('Something went wrong.'+error.name))};return res.status(Codes.OK).json(result)})}

src\controllers\Order\index.ts

export{createNewOrder}from'./createNewOrder';export{getOrder}from'./getOrder';export{deleteOrder}from'./deleteOrder';export{updateOrder}from'./updateOrder';export{getAllOrders}from''

src\controllers\Order\updateOrder.ts

import{Types}from'mongoose';import{Request,Response}from'express';import{Order}from'../../models';import{Status,ErrorMessageAnswer,Codes}from'../../types';import{createErrorMessage}from'../../helpers/messages';import{getValidParamsWithCheckAll}from'../../helpers/getValidParams';type getQueryParamsOk={status:Status.OK;value:{id:Types.ObjectId;user:Types.ObjectId|undefined;product:Types.ObjectId|undefined;count:number|undefined;productPrice:number|undefined;address:string|undefined;status:string|undefined}};const getQueryParams=(req:Request):ErrorMessageAnswer|getQueryParamsOk=>{const{id,user,product,count,productPrice,address,status}=req.query;let resultId:undefined|Types.ObjectId=undefined;let resultUser:undefined|Types.ObjectId=undefined;let resultProduct:undefined|Types.ObjectId=undefined;if(typeof id!=='string'){return createErrorMessage('Id must be string')};if(!Types.ObjectId.isValid(id)){return createErrorMessage('Id string must be ObjectId.isValid')};resultId=new Types.ObjectId(id);if(user!==undefined){if(typeof user!=='string'){return createErrorMessage('user must be undefined | string')};if(!Types.ObjectId.isValid(user)){return createErrorMessage('user string must be ObjectId.isValid')};resultUser=new Types.ObjectId(user)};if(product!==undefined){if(typeof product!=='string'){return createErrorMessage('product must be undefined | string')};if(!Types.ObjectId.isValid(product)){return createErrorMessage('product string must be ObjectId.isValid')};resultProduct=new Types.ObjectId(product)};let resultCount:undefined|number=undefined;let resultProductPrice:undefined|number=undefined;if(count!==undefined){if(Number(count)===NaN){return createErrorMessage('address must be string')};resultCount=Number(count)};if(productPrice!==undefined){if(Number(productPrice)===NaN){return createErrorMessage('address must be string')};resultProductPrice=Number(productPrice)};if(address!==undefined){if(typeof address!=='string'){return createErrorMessage('address must be string')}};if(status!==undefined){if(typeof status!=='string'){return createErrorMessage('status must be string')}};return{status:Status.OK,value:{id:resultId,user:resultUser,product:resultProduct,count:resultCount,productPrice:resultProductPrice,address,status,},}};export const updateOrder=async(req:Request,res:Response)=>{const params=getQueryParams(req);if(params.status===Status.ERROR){return res.status(Codes.ERROR).json(params)};const{id}=params.value;const updatedParams=getValidParamsWithCheckAll(params.value);Order.updateOne({\_id:id},updatedParams,null,error=>{if(error){return res.status(Codes.ERROR).json(createErrorMessage('Something went wrong.'+error.name))};Order.findOne({\_id:id}).exec((findError,findResult)=>{if(findError){return res.status(Codes.ERROR).json(createErrorMessage('Something went wrong.'+findError.name))};return res.status(Codes.OK).json(findResult)})})}

src\controllers\Producer\createNewProducer.ts

import mongoose,{Types}from'mongoose';import{Request,Response}from'express';import{Producer}from'../../models';import{Status,ErrorMessageAnswer,Codes}from'../../types';import{createErrorMessage}from'../../helpers/messages';import{getValidParamsWithCheckID}from'../../helpers/getValidParams';type getQueryParamsOk={status:Status.OK;value:{id:Types.ObjectId|undefined;title:string;description:string;website:string}};const getQueryParams=(body:Request['body']):ErrorMessageAnswer|getQueryParamsOk=>{const{\_id,title,description,website}=body;let resultId:undefined|Types.ObjectId=undefined;if(\_id!==undefined){if(typeof \_id!=='string'){return createErrorMessage('Id must be undefined | string')};if(!Types.ObjectId.isValid(\_id)){return createErrorMessage('Id string must be ObjectId isValid')};resultId=new Types.ObjectId(\_id)};if(typeof title!=='string'){return createErrorMessage('Title must be string')};if(typeof description!=='string'){return createErrorMessage('Description must be string')};if(typeof website!=='string'){return createErrorMessage('Website must be string')};return{status:Status.OK,value:{id:resultId,title,description,website},}};export const createNewProducer=async(req:Request,res:Response)=>{const params=getQueryParams(req.body);if(params.status===Status.ERROR){return res.status(Codes.ERROR).json(params)};const{id,title,description,website}=params.value;const newProducerParams=getValidParamsWithCheckID({id,title,description,website,});new Producer(newProducerParams).save((error,result)=>{if(error){return res.status(Codes.ERROR).json(createErrorMessage('Something went wrong.'+error.name))};return res.status(Codes.OK).json(result)})}

src\controllers\Producer\deleteProducer.ts

import{Types}from'mongoose';import{Request,Response}from'express';import{Producer}from'../../models';import{Codes}from'../../types';import{createErrorMessage,createSuccessMessage,}from'../../helpers/messages';export const deleteProducer=(req:Request,res:Response)=>{const id=req.params.\_id;if(typeof id!=='string'){return createErrorMessage('Id must be string')};if(!Types.ObjectId.isValid(id)){return createErrorMessage('Id string must be ObjectId.isValid')};const idAsObjectId=new Types.ObjectId(id);Producer.deleteOne({\_id:idAsObjectId},error=>{if(error){return res.status(Codes.ERROR).json(createErrorMessage('Something went wrong.'+error.name))};return res.status(Codes.OK).json(createSuccessMessage(`Producer ${id}successfully deleted`))})}

src\controllers\Producer\getAllProducers.ts

import{Request,Response}from'express';import{Producer}from'../../models';import{Codes}from'../../types';import{createErrorMessage}from'../../helpers/messages';export const getAllProducers=(req:Request,res:Response)=>{Producer.find({},{\_\_v:0}).exec((error,result)=>{if(error){return res.status(Codes.ERROR).json(createErrorMessage('Something went wrong.'+error.name))};return res.status(Codes.OK).json(result)})}

src\controllers\Producer\getProducer.ts

import{Types}from'mongoose';import{Request,Response}from'express';import{Producer}from'../../models';import{Status,ErrorMessageAnswer,Codes}from'../../types';import{createErrorMessage}from'../../helpers/messages';import{getValidParamsWithCheckAll}from'../../helpers/getValidParams';type getQueryParamsOk={status:Status.OK;value:{id:Types.ObjectId|undefined;title:string|undefined;description:string|undefined;website:string|undefined}};const getQueryParams=(req:Request):ErrorMessageAnswer|getQueryParamsOk=>{const{id,title,description,website}=req.query;let resultId:undefined|Types.ObjectId=undefined;if(id!==undefined){if(typeof id!=='string'){return createErrorMessage('Id must be undefined | string')};if(!Types.ObjectId.isValid(id)){return createErrorMessage('Id string must be ObjectId.isValid')};resultId=new Types.ObjectId(id)};if(title!==undefined){if(typeof title!=='string'){return createErrorMessage('Title must be string')}};if(description!==undefined){if(typeof description!=='string'){return createErrorMessage('Description must be string')}};if(website!==undefined){if(typeof website!=='string'){return createErrorMessage('Website must be string')}};return{status:Status.OK,value:{id:resultId,title,description,website},}};export const getProducer=async(req:Request,res:Response)=>{const params=getQueryParams(req);if(params.status===Status.ERROR){return res.status(Codes.ERROR).json(params)};const{id,title,description,website}=params.value;const findParams=getValidParamsWithCheckAll({id,title,description,website,});Producer.findOne(findParams,{\_\_v:0}).exec((error,result)=>{if(error){return res.status(Codes.ERROR).json(createErrorMessage('Something went wrong.'+error.name))};return res.status(Codes.OK).json(result)})}

src\controllers\Producer\index.ts

export{createNewProducer}from'./createNewProducer';export{deleteProducer}from'./deleteProducer';export{getProducer}from'./getProducer';export{updateProducer}from'./updateProducer';export{getAllProducers}from''

src\controllers\Producer\updateProducer.ts

import{Types}from'mongoose';import{Request,Response}from'express';import{Producer}from'../../models';import{Status,ErrorMessageAnswer,Codes}from'../../types';import{createErrorMessage}from'../../helpers/messages';import{getValidParamsWithCheckAll}from'../../helpers/getValidParams';type getQueryParamsOk={status:Status.OK;value:{id:Types.ObjectId;title:string|undefined;description:string|undefined;website:string|undefined}};const getQueryParams=(req:Request):ErrorMessageAnswer|getQueryParamsOk=>{const{id,title,description,website}=req.query;if(typeof id!=='string'){return createErrorMessage('Id must be undefined | string')};if(!Types.ObjectId.isValid(id)){return createErrorMessage('Id string must be ObjectId.isValid')};const resultId=new Types.ObjectId(id);if(title!==undefined){if(typeof title!=='string'){return createErrorMessage('Title must be string')}};if(description!==undefined){if(typeof description!=='string'){return createErrorMessage('Description must be string')}};if(website!==undefined){if(typeof website!=='string'){return createErrorMessage('Website must be string')}};return{status:Status.OK,value:{id:resultId,title,description,website},}};export const updateProducer=async(req:Request,res:Response)=>{const params=getQueryParams(req);if(params.status===Status.ERROR){return res.status(Codes.ERROR).json(params)};const{id,title,description,website}=params.value;const updatedParams=getValidParamsWithCheckAll({id,title,description,website,});Producer.updateOne({\_id:id},updatedParams,null,error=>{if(error){return res.status(Codes.ERROR).json(createErrorMessage('Something went wrong.'+error.name))};Producer.findOne({\_id:id}).exec((findError,findResult)=>{if(findError){return res.status(Codes.ERROR).json(createErrorMessage('Something went wrong.'+findError.name))};return res.status(Codes.OK).json(findResult)})})}

src\controllers\Product\ProductPost.ts

import mongoose from'mongoose';import{Category,Producer,ProductsAllInfo,Type,User}from'../../models';export const productPostController=async(data:any)=>{console.log(data);const notValidateFields={title:{ua:data.titleUA,en:data.titleEN,},description:{ua:data.descriptionUA,en:data.descriptionEN,},price:data.price,stars:data.stars,novelty:data.novelty,discount:data.discount,discountValue:data.discountValue,discountPreviousValue:data.discountPreviousValue,};const queryType=data.type;const categoryType=data.category;const producerType=data.producer;const creatorQuery=data.creator;const typeValue=await typeValidate(queryType);const categoryValue=await categoryValidate(categoryType);const producerValue=await producerValidate(producerType);const allInfoValue=await allInfoCreate();const creatorValue=await creatorValidate(creatorQuery);if(typeValue.status==='error'){return typeValue};if(categoryValue.status==='error'){return categoryValue};if(producerValue.status==='error'){return producerValue};if(creatorValue.status==='error'){return creatorValue};return{...notValidateFields,type:typeValue.value,category:categoryValue.value,producer:producerValue.value,creator:creatorValue.value,allInfo:allInfoValue.value,}};const typeValidate=async(type:any)=>{if(typeof type!=='string'){return{status:'error',message:'Invalid type',}};if(type.includes(',')){const typeArray=type.split(',');const equals=typeArray.map(item=>{return{title:item}});const result=await Type.find({$or:equals});if(result.length!==typeArray.length){return{status:'error',message:'Invalid type',}};return{status:'ok',value:result.map(item=>item.\_id),}};const result=await Type.findOne({title:type});if(result===null){return{status:'error',message:'Invalid type',}};return{status:'ok',value:result.\_id,}};const categoryValidate=async(category:any)=>{if(typeof category!=='string'){return{status:'error',message:'Invalid category',}};if(category.includes(',')){const categoryArray=category.split(',');const equals=categoryArray.map(item=>{return{title:item}});const result=await Category.find({$or:equals});if(result.length!==categoryArray.length){return{status:'error',message:'Invalid category',}};return{status:'ok',value:result.map(item=>item.\_id),}};const result=await Category.findOne({title:category});if(result===null){return{status:'error',message:'Invalid category',}};return{status:'ok',value:result.\_id,}};const producerValidate=async(value:any)=>{if(typeof value!=='string'){return{status:'error',message:'Invalid producer',}};const result=await Producer.findOne({title:value});if(result===null){return{status:'error',message:'Invalid producer',}};return{status:'ok',value:result.\_id,}};const allInfoCreate=async()=>{const allInfo=await new ProductsAllInfo({description:'description',characteristics:'characteristics',reviews:[],}).save();return{status:'ok',value:allInfo.\_id,}};const creatorValidate=async(value:any)=>{if(typeof value!=='string'){return{status:'error',message:'Invalid creator',}};const result=await User.findOne({\_id:new mongoose.Types.ObjectId(value)});if(result===null){return{status:'error',message:'Invalid creator',}};return{status:'ok',value:result.\_id,}}

src\controllers\Type\createNewType.ts

import mongoose,{Types}from'mongoose';import{Request,Response}from'express';import{Type}from'../../models';import{Status,ErrorMessageAnswer,Codes}from'../../types';import{createErrorMessage}from'../../helpers/messages';import{getValidParamsWithCheckID}from'../../helpers/getValidParams';type getQueryParamsOk={status:Status.OK;value:{id:Types.ObjectId|undefined;title:string;description:string}};const getQueryParams=(body:Request['body']):ErrorMessageAnswer|getQueryParamsOk=>{const{\_id,title,description}=body;let resultId:undefined|Types.ObjectId=undefined;if(\_id!==undefined){if(typeof \_id!=='string'){return createErrorMessage('Id must be undefined | string')};if(!Types.ObjectId.isValid(\_id)){return createErrorMessage('Id string must be ObjectId isValid')};resultId=new Types.ObjectId(\_id)};if(typeof title!=='string'){return createErrorMessage('Title must be string')};if(typeof description!=='string'){return createErrorMessage('Description must be string')};return{status:Status.OK,value:{id:resultId,title,description},}};export const createNewType=async(req:Request,res:Response)=>{const params=getQueryParams(req.body);if(params.status===Status.ERROR){return res.status(Codes.ERROR).json(params)};const{id,title,description}=params.value;const newCategoryParams=getValidParamsWithCheckID({id,title,description,});new Type(newCategoryParams).save((error,result)=>{if(error){return res.status(Codes.ERROR).json(createErrorMessage('Something went wrong.'+error.name))};return res.status(Codes.OK).json(result)})}

src\controllers\Type\deleteType.ts

import{Types}from'mongoose';import{Request,Response}from'express';import{Type}from'../../models';import{Codes}from'../../types';import{createErrorMessage,createSuccessMessage,}from'../../helpers/messages';export const deleteType=(req:Request,res:Response)=>{const id=req.params.\_id;if(typeof id!=='string'){return createErrorMessage('Id must be string')};if(!Types.ObjectId.isValid(id)){return createErrorMessage('Id string must be ObjectId.isValid')};const idAsObjectId=new Types.ObjectId(id);Type.deleteOne({\_id:idAsObjectId},error=>{if(error){return res.status(Codes.ERROR).json(createErrorMessage('Something went wrong.'+error.name))};return res.status(Codes.OK).json(createSuccessMessage(`Type ${id}successfully deleted`))})}

src\controllers\Type\getAllTypes.ts

import{Request,Response}from'express';import{Type}from'../../models';import{Codes}from'../../types';import{createErrorMessage}from'../../helpers/messages';export const getAllTypes=(req:Request,res:Response)=>{Type.find({},{\_\_v:0}).exec((error,result)=>{if(error){return res.status(Codes.ERROR).json(createErrorMessage('Something went wrong.'+error.name))};return res.status(Codes.OK).json(result)})}

src\controllers\Type\getType.ts

import{Types}from'mongoose';import{Request,Response}from'express';import{Type}from'../../models';import{Status,ErrorMessageAnswer,Codes}from'../../types';import{createErrorMessage}from'../../helpers/messages';import{getValidParamsWithCheckAll}from'../../helpers/getValidParams';type getQueryParamsOk={status:Status.OK;value:{id:Types.ObjectId|undefined;title:string|undefined;description:string|undefined}};const getQueryParams=(req:Request):ErrorMessageAnswer|getQueryParamsOk=>{const{id,title,description}=req.query;let resultId:undefined|Types.ObjectId=undefined;if(id!==undefined){if(typeof id!=='string'){return createErrorMessage('Id must be undefined | string')};if(!Types.ObjectId.isValid(id)){return createErrorMessage('Id string must be ObjectId.isValid')};resultId=new Types.ObjectId(id)};if(title!==undefined){if(typeof title!=='string'){return createErrorMessage('Title must be string')}};if(description!==undefined){if(typeof description!=='string'){return createErrorMessage('Description must be string')}};return{status:Status.OK,value:{id:resultId,title,description},}};export const getType=async(req:Request,res:Response)=>{const params=getQueryParams(req);if(params.status===Status.ERROR){return res.status(Codes.ERROR).json(params)};const{id,title,description}=params.value;const findParams=getValidParamsWithCheckAll({id,title,description});Type.findOne(findParams,{\_\_v:0}).exec((error,result)=>{if(error){return res.status(Codes.ERROR).json(createErrorMessage('Something went wrong.'+error.name))};return res.status(Codes.OK).json(result)})}

src\controllers\Type\index.ts

export{createNewType}from'./createNewType';export{getType}from'./getType';export{deleteType}from'./deleteType';export{updateType}from'./updateType';export{getAllTypes}from''

src\controllers\Type\updateType.ts

import{Types}from'mongoose';import{Request,Response}from'express';import{Type}from'../../models';import{Status,ErrorMessageAnswer,Codes}from'../../types';import{createErrorMessage}from'../../helpers/messages';import{getValidParamsWithCheckAll}from'../../helpers/getValidParams';type getQueryParamsOk={status:Status.OK;value:{id:Types.ObjectId;title:string|undefined;description:string|undefined}};const getQueryParams=(req:Request):ErrorMessageAnswer|getQueryParamsOk=>{const{id,title,description}=req.query;if(typeof id!=='string'){return createErrorMessage('Id must be undefined | string')};if(!Types.ObjectId.isValid(id)){return createErrorMessage('Id string must be ObjectId.isValid')};const resultId=new Types.ObjectId(id);if(title!==undefined){if(typeof title!=='string'){return createErrorMessage('Title must be string')}};if(description!==undefined){if(typeof description!=='string'){return createErrorMessage('Description must be string')}};return{status:Status.OK,value:{id:resultId,title,description},}};export const updateType=async(req:Request,res:Response)=>{const params=getQueryParams(req);if(params.status===Status.ERROR){return res.status(Codes.ERROR).json(params)};const{id,title,description}=params.value;const updatedParams=getValidParamsWithCheckAll({id,title,description});Type.updateOne({\_id:id},updatedParams,null,error=>{if(error){return res.status(Codes.ERROR).json(createErrorMessage('Something went wrong.'+error.name))};Type.findOne({\_id:id}).exec((findError,findResult)=>{if(findError){return res.status(Codes.ERROR).json(createErrorMessage('Something went wrong.'+findError.name))};return res.status(Codes.OK).json(findResult)})})}

src\helpers\getValidParams.ts

import{Types}from'mongoose';export function getValidParamsWithCheckID<T extends{id:Types.ObjectId|undefined}>(value:T):Omit<T,'id'>&{\_id?:Types.ObjectId}{const result=Object.keys(value).reduce((acc,current)=>{if(current!=='id'){return{...acc,[current]:value[current]}};if(value[current]!==undefined){return{...acc,\_id:value[current]}};return acc},{})as Omit<T,'id'>&{\_id?:Types.ObjectId};return result};export function getValidParamsWithCheckAll<T extends{id:Types.ObjectId|undefined}>(value:T){const checkedId=getValidParamsWithCheckID(value);const res=Object.fromEntries(Object.entries(checkedId).filter(([key,value])=>value!==undefined));return res}

src\helpers\messages.ts

import{Status,ErrorMessageAnswer,SuccessMessageAnswer}from'../types';export const createErrorMessage=(message:string):ErrorMessageAnswer=>({status:Status.ERROR,message:message,});export const createSuccessMessage=(message:string):SuccessMessageAnswer=>({status:Status.OK,message:message,})

src\models\AccountLevel.ts

import{Schema,model}from'mongoose';const accountLevelSchema=new Schema({level:{type:Number,required:true},description:{type:String,required:true},});const AccountLevel=model('AccountLevel',accountLevelSchema);export{AccountLevel}

src\models\Category.ts

import{Schema,model}from'mongoose';const categorySchema=new Schema({title:{type:String,required:true},description:{type:String,required:true},});const Category=model('Category',categorySchema);export{Category}

src\models\index.ts

export{Type}from'./Type';export{Review}from'./Review';export{ProductsAllInfo}from'./ProductsAllInfo';export{Category}from'./Category';export{Producer}from'./Producer';export{AccountLevel}from'./AccountLevel';export{User}from'./User';export{Order}from'./Order';export{Product}from''

src\models\Order.ts

import{Schema,model}from'mongoose';import{User}from'.';import{Product}from'./Product';const orderSchema=new Schema({user:{type:Schema.Types.ObjectId,ref:User,required:true},product:{type:Schema.Types.ObjectId,ref:Product,required:true},count:{type:Number,required:true},productPrice:{type:Number,required:true},address:{type:String,required:true},status:{type:String,required:true},},{timestamps:true,});const Order=model('Order',orderSchema);export{Order}

src\models\Producer.ts

import{Schema,model}from'mongoose';const producerSchema=new Schema({title:{type:String,required:true},description:{type:String,required:true},website:{type:String,required:true},},{timestamps:true,});const Producer=model('Producer',producerSchema);export{Producer}

src\models\Product.ts

import{Schema,model}from'mongoose';import{Category,Producer,ProductsAllInfo,Type,User}from'.';const productSchema=new Schema({title:{en:{type:String,required:true},ua:{type:String,required:true},},description:{en:{type:String,required:true},ua:{type:String,required:true},},price:{type:Number,required:true},stars:{type:Number,required:true},image:{type:Buffer,contentType:String,required:true},type:{type:[Schema.Types.ObjectId],ref:Type,required:true},category:{type:[Schema.Types.ObjectId],ref:Category,required:true},producer:{type:Schema.Types.ObjectId,ref:Producer,required:true},novelty:{type:Boolean,default:false,required:true},discount:{type:Boolean,default:false,required:true},discountValue:{type:Number,default:-1,required:true},discountPreviousPrice:{type:Number,default:-1,required:true},allInfo:{type:Schema.Types.ObjectId,ref:ProductsAllInfo,required:true,},creator:{type:Schema.Types.ObjectId,ref:User,required:true},},{timestamps:true,});const Product=model('Product',productSchema);export{Product}

src\models\ProductsAllInfo.ts

import{Schema,model}from'mongoose';import{Review}from'.';const productsAllInfoSchema=new Schema({description:{type:String,required:true},characteristics:{type:String,required:true},reviews:{type:[Schema.Types.ObjectId],ref:Review},},{timestamps:true,});const ProductsAllInfo=model('ProductsAllInfo',productsAllInfoSchema);export{ProductsAllInfo}

src\models\Review.ts

import{Schema,model}from'mongoose';import{User}from'./User';const rewiewsSchema=new Schema({author:{type:Schema.Types.ObjectId,ref:User},comment:{type:String,required:true},pluses:{type:String,required:true},minuses:{type:String,required:true},rating:{type:Number,required:true},},{timestamps:true,});const Review=model('Review',rewiewsSchema);export{Review}

src\models\Type.ts

import{Schema,model}from'mongoose';const typeSchema=new Schema({title:{type:String,required:true},description:{type:String,required:true},});const Type=model('Type',typeSchema);export{Type}

src\models\User.ts

import{Schema,model}from'mongoose';import{AccountLevel}from'./AccountLevel';const userSchema=new Schema({surname:{type:String,required:true},name:{type:String,required:true},email:{type:String,required:true},phone:{type:String,required:true},password:{type:String,required:true},country:{type:String,required:true},city:{type:String,required:true},zipCode:{type:Number,required:true},address:{type:String,required:true},accountAdminLevel:{type:Schema.Types.ObjectId,ref:AccountLevel,required:true,},},{timestamps:true,});const User=model('User',userSchema);export{User}

src\routes\accountLevel.ts

import Router,{Request,Response}from'express';import{createNewAccountLevel,deleteAccountLevel,getAccountLevel,updateAccountLevel,getAllAccountLevels,}from'../controllers/AccountLevel';const AccountLevelRouter=Router();AccountLevelRouter.get('/accountLevel/all',getAllAccountLevels);AccountLevelRouter.get('/accountLevel',getAccountLevel);AccountLevelRouter.post('/accountLevel',createNewAccountLevel);AccountLevelRouter.delete('/accountLevel/:\_id',deleteAccountLevel);AccountLevelRouter.put('/accountLevel',updateAccountLevel);export{AccountLevelRouter}

src\routes\categories.ts

import Router,{Request,Response}from'express';import{createNewCategory,deleteCategory,getAllCategories,getCategories,updateCategory,}from'../controllers/Category';const CategoriesRouter=Router();CategoriesRouter.use((req:Request,res:Response,next)=>{console.log('[Server]: Time = ',Date.now());next()});CategoriesRouter.get('/categories/all',getAllCategories);CategoriesRouter.get('/categories',getCategories);CategoriesRouter.post('/categories',createNewCategory);CategoriesRouter.put('/categories',updateCategory);CategoriesRouter.delete('/categories/:\_id',deleteCategory);export{CategoriesRouter}

src\routes\home.ts

import Router,{Request,Response}from'express';const router=Router();router.use((req:Request,res:Response,next)=>{console.log('[Server]: Time = ',Date.now());next()});router.get('/',(req:Request,res:Response)=>{res.status(200).json({page:'home',type:'get'})});router.post('/',(req:Request,res:Response)=>{res.status(200).json({page:'home',type:'post'})});export{router as HomeRouter}

src\routes\index.ts

import Router,{Request,Response}from'express';import{User}from'../models';export{HomeRouter}from'./home';export{AccountLevelRouter}from'./accountLevel';export{CategoriesRouter}from'./categories';export{ProducerRouter}from'./producers';export{TypesRouter}from'./types';export{OrdersRouter}from''

src\routes\orders.ts

import Router,{Request,Response}from'express';import{getAllOrders,getOrder,createNewOrder,updateOrder,deleteOrder,}from'../controllers/Order';const OrdersRouter=Router();OrdersRouter.get('/order/all',getAllOrders);OrdersRouter.get('/order',getOrder);OrdersRouter.post('/order',createNewOrder);OrdersRouter.put('/order',updateOrder);OrdersRouter.delete('/order/:\_id',deleteOrder);export{OrdersRouter}

src\routes\producers.ts

import Router,{Request,Response}from'express';import{createNewProducer,deleteProducer,getProducer,updateProducer,getAllProducers,}from'../controllers/Producer';const ProducerRouter=Router();ProducerRouter.get('/producer/all',getAllProducers);ProducerRouter.get('/producer',getProducer);ProducerRouter.post('/producer',createNewProducer);ProducerRouter.delete('/producer/:\_id',deleteProducer);ProducerRouter.put('/producer',updateProducer);export{ProducerRouter}

src\routes\types.ts

import Router,{Request,Response}from'express';import{createNewType,deleteType,getType,updateType,getAllTypes,}from'../controllers/Type';const TypesRouter=Router();TypesRouter.get('/type/all',getAllTypes);TypesRouter.get('/type',getType);TypesRouter.post('/type',createNewType);TypesRouter.delete('/type/:\_id',deleteType);TypesRouter.put('/type',updateType);export{TypesRouter}

src\types\index.ts

export\*from''

src\types\main.ts

export enum Status{OK='ok',ERROR='error',};export enum Codes{OK=200,ERROR=500,};type MessageAnswer<T extends Status>={status:T;message:string};export type SuccessMessageAnswer=MessageAnswer<Status.OK>export type ErrorMessageAnswer=MessageAnswer<Status.ERROR>

src\utils\initializeDatabase\AccountLevel.ts

import mongoose,{Schema,ObjectId}from'mongoose';import{AccountLevel}from'../../models';const initializeDatabaseAccountLevel=async()=>{await AccountLevel.deleteMany({});const createAccountLevelInitialize=(\_id:mongoose.Types.ObjectId,level:number,description:string)=>{return new AccountLevel({\_id:\_id,level:level,description:description,}).save(error=>{if(error){console.error(`AdminLevel ${level}error`,error);return};console.log(`AdminLevel ${level}has been added`)})};createAccountLevelInitialize(new mongoose.Types.ObjectId('63838c77675cff96068e1823'),0,'Default');createAccountLevelInitialize(new mongoose.Types.ObjectId('63838c77675cff96068e1824'),1,'Moderator');createAccountLevelInitialize(new mongoose.Types.ObjectId('63838c77675cff96068e1825'),2,'Admin')};export{initializeDatabaseAccountLevel}

src\utils\initializeDatabase\Category.ts

import mongoose,{Schema,ObjectId}from'mongoose';import{Category}from'../../models';const initializeDatabaseCategory=async()=>{await Category.deleteMany({});const createCategoryInitialize=(\_id:mongoose.Types.ObjectId,title:string,description:string)=>{return new Category({\_id:\_id,title:title,description:description,}).save(error=>{if(error){console.error(`Category ${title}error`,error);return};console.log(`Category ${title}has been added`)})};createCategoryInitialize(new mongoose.Types.ObjectId('638387eb6f33fac689be5e81'),'Care','Care');createCategoryInitialize(new mongoose.Types.ObjectId('638387eb6f33fac689be5e82'),'Walk','Walk');createCategoryInitialize(new mongoose.Types.ObjectId('638387eb6f33fac689be5e83'),'Health','Health');createCategoryInitialize(new mongoose.Types.ObjectId('638387eb6f33fac689be5e84'),'Toys','Toys');createCategoryInitialize(new mongoose.Types.ObjectId('638387eb6f33fac689be5e85'),'SleepingPlaces','SleepingPlaces');createCategoryInitialize(new mongoose.Types.ObjectId('638387eb6f33fac689be5e80'),'Bowls','Bowls')};export{initializeDatabaseCategory}

src\utils\initializeDatabase\index.ts

import{initializeDatabaseCategory}from'./Category';import{initializeDatabaseType}from'./Type';import{initializeDatabaseAccountLevel}from'./AccountLevel';const initializeDataBase=()=>{initializeDatabaseType();initializeDatabaseCategory();initializeDatabaseAccountLevel()};export{initializeDataBase}

src\utils\initializeDatabase\Type.ts

import mongoose,{Schema,ObjectId}from'mongoose';import{Type}from'../../models';const initializeDatabaseType=()=>{Type.deleteMany({},()=>{});const createTypeInitialize=(\_id:mongoose.Types.ObjectId,title:string,description:string)=>{return new Type({\_id:\_id,title:title,description:description,}).save(error=>{if(error){console.error(`Type ${title}error`,error);return};console.log(`Type ${title}has been added`)})};createTypeInitialize(new mongoose.Types.ObjectId('6383529877588dc76e74919c'),'Cats','Cats');createTypeInitialize(new mongoose.Types.ObjectId('6383529877588dc76e74919d'),'Dogs','Dogs')};export{initializeDatabaseType}