

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВІДОКРЕМЛЕНИЙ СТРУКТУРНИЙ ПІДРОЗДІЛ
«ЗАПОРІЗЬКИЙ ЕЛЕКТРОТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ
НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ «ЗАПОРІЗЬКА ПОЛІТЕХНІКА»

Циклова комісія спеціальності
121 Інженерія програмного забезпечення
спеціалізація «Розробка програмного забезпечення»

РОЗРОБКА БАГАТОШАРОВОЇ
КЛІЄНТ-СЕРВЕРНОЇ АРХІТЕКТУРИ

Пояснювальна записка до дипломної роботи

121.00.29.01 ПЗ

Керівник роботи

Ніна БАБЕНКО

Консультанти

зі спеціальної частини

Ольга СЛАДКОВА

з економіки

Тетяна ПИЛИПЕНКО

з охорони праці

Жанна ФЕДОРІНА

Нормоконтролер

Жанна ФЕДОРІНА

Студент групи РПЗ 19 ²/9

Іван ЩЕДРОВСЬКИЙ

2023

РЕФЕРАТ

Пояснювальна записка до дипломної роботи містить 194 сторінок, 68 рисунків, 11 таблиць, 1 додаток, 82 джерел.

Пояснювальна записка складається з шести розділів.

Розділ «Опис предметної області» містить основні поняття та алгоритм роботи сервісу.

Розділ «Постанова завдання» включає мету створення та функції програми, вимоги до проєктованої системи, вимоги до надійності програмного продукту, умови роботи програми.

Розділ «Програмування» містить обґрунтування вибору середовища розробки та функціонування системи, основні рішення щодо реалізації компонентів системи.

Розділ «Методика роботи користувача з системою» включає в себе керівництво для програміста та оператора.

«Організаційно-економічний розділ» містить планування розробки програмного продукту, визначення витрат на розробку програмного продукту та оцінку ефективності проєкту.

Розділ «Охорона праці користувачів комп'ютерів» включає правові аспекти охорони праці користувачів комп'ютерів, розглянуті питання електробезпеки, пожежної безпеки в приміщеннях з персональними комп'ютерами, дана загальна характеристика надзвичайних ситуацій.

СТАТИСТИКА, ПЛАНУВАННЯ, ANGULAR, NESTJS, JAVASCRIPT, TYPESCRIPT, MONGODB, NOSQL, GOOGLE IDENTITY, JSON WEB TOKENS, AUTHORIZATION, API, PWA, CORS, ANGULAR MATERIAL, ANGULAR CLI, DRAG-N-DROP, ПАТЕРНИ ПРОЄКТУВАННЯ, КЕРІВНИЦТВО ПРОГРАМІСТА, ПЛАНУВАННЯ РОЗРОБКИ, ОХОРОНА ПРАЦІ.

ЗМІСТ

Вступ.....	5
1 Опис предметної області	6
1.1 Основні поняття	6
1.2 Основний алгоритм.....	16
2 Постанова завдання.....	17
2.1 Мета створення програми	17
2.2 Функції програми	17
2.3 Вимоги до проектованої системи	18
2.4 Вимоги до надійності.....	19
2.5 Умови роботи програми	19
2.6 Умови розповсюдження програми	21
3 Програмування	23
3.1 Обґрунтування вибору середовища розробки системи	23
3.2 Обґрунтування вибору середовища функціонування системи	29
3.3 Основні рішення щодо реалізації компонентів системи.....	33
3.3.1 Використовувані моделі даних	33
3.3.2 Структурна схема програми.....	41
3.3.3 Розробка модулів системи.....	42
4 Методика роботи користувача з системою	83
4.1 Керівництво програміста.....	83
4.1.1 Призначення і умови використання програми.....	83
4.1.2 Характеристики програми.....	83
4.1.3 Звертання до програми	84
4.1.4 Вхідні і вихідні дані	85
4.2 Керівництво оператора	85
4.2.1 Призначення і умови використання програми.....	85
4.2.2 Виконання програми.....	86

5 Організаційно – економічний розділ.....	96
5.1 Планування розробки програмного продукту.....	96
5.2 Розрахунок витрат на розробку програмного продукту	98
5.2.1 Складання кошторису витрат на розробку.....	98
5.2.2 Розрахунок собівартості програмного продукту	105
5.3 Оцінка ефективності проєкту	108
6 Охорона праці користувачів комп'ютерів.....	112
6.1 Правове забезпечення заходів щодо охорони праці користувачів комп'ютерів.....	112
6.2 Електробезпека та пожежна безпека в приміщеннях з персональними комп'ютерами.	121
6.3 Причини виникнення, загальна характеристика та класифікація надзвичайних ситуацій	127
Висновки	134
Перелік джерел посилання	135
Додаток А (обов'язковий) - Текст програми.....	142

ВСТУП

Завдяки прогресу та розвитку інтернету обсяг доступної інформації для людей значно збільшився. Раніше отримання певної інформації було нелегким завданням, але зараз з легкістю можна знайти будь-яку інформацію через пошукові мережі та доступ до всесвітньої мережі даних. Інформація стала відкритою для усіх бажаючих, що призвело до значного збільшення її обсягу. Для зберігання великої кількості даних потрібен зручний сервіс, який зможе це робити

Не так давно вчені підраховали, що сучасна людина за тиждень отримує стільки інформації, скільки людина середньовіччя отримувала за все життя. Людська психіка має певні обмеження. Експериментально доведено, що мозок звичайної людини здатен сприймати і безпомилково обробляти інформацію зі швидкістю не більше 25 біт на секунду (в одному слові середньої довжини міститься якраз 25 біт). При такій швидкості поглинання інформації людина за життя може прочитати не більше трьох тисяч книг. І то – за умови, що буде щодня освоювати по 50 сторінок. Мало того, що ми не встигаємо вивчити велику частину інформації, яка накопичується, вона ще й швидко старіє і вимагає заміни. Вперше над цим фактом задумалися вчені у 70-х роках минулого століття. Тоді і почав використовуватись термін «інформаційний вибух». [1]

Мета цієї дипломної роботи полягає в створенні багатоварової клієнт-серверної архітектури, яка спрощує та удосконалює обробку великих обсягів інформації, зокрема, за рахунок зручної маніпуляції статистичними даними.

1 ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Основні поняття

Предмет науки (предметна область) – це ті сторони, зв'язки, відношення об'єкта, які вивчаються даною наукою [2]

Мета цієї дипломної роботи полягає в створенні багатошарової клієнт-серверної архітектури, яка спрощує та удосконалює обробку великих обсягів інформації, зокрема, за рахунок зручної маніпуляції статистичними даними.

Розробка програмного забезпечення (ПЗ) – це вид діяльності та процес, спрямований на створення та підтримку працездатності, якості та надійності ПЗ, використовуючи технології, методологію та практики з інформатики, керування проектами, математики, інженерії та інших областей знання. Як і інші традиційні інженерні дисципліни, розробка ПЗ має справу з проблемами якості, вартості та надійності. Деякі програми містять мільйони рядків вихідного коду, які, як очікується, повинні правильно виконуватися в умовах, що змінюються. Складність ПЗ порівнянна зі складністю найбільш складних сучасних машин (таких, наприклад, як літаки). [3]

Елементами сервісу введення статистики є: категорії, статичні дані, графік, авторизація, активні сесії користувача,

Категорії в сервісі служать для об'єднання даних, які користувач може ввести в систему для зручного збереження та роботою з даними різного виду, таких як, наприклад, різні види спортивних вправ або ж різні сфери

Статичні дані в сервісі - дані, які користувач може вводити в систему, наприклад, кількість відвідувачів сайту за день або продажі за тиждень, обираючи для них якусь категорію.

Графік в сервісі - візуальне представлення статистичних даних у вигляді графіка з різними режимами відображення. Графіки допомагають користувачам аналізувати дані і зробити висновки про тенденції і зміни у певному періоді часу.

Автентифікація - це процес перевірки особистості користувача. Технологія перевірки автентичності забезпечує контроль доступу для систем, перевіряючи, чи облікові дані користувача збігаються з обліковими даними в базі даних авторизованих користувачів або на сервері автентифікації даних [4]

Активні сесії користувача - це перелік сеансів, що знаходяться у відкритому стані на різних пристроях, що користується користувач. Кожна сесія відображається відповідно до типу пристрою, часу та інших параметрів. Це дозволяє користувачам управляти своїм доступом до сервісу та контролювати власну безпеку в Інтернеті.

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

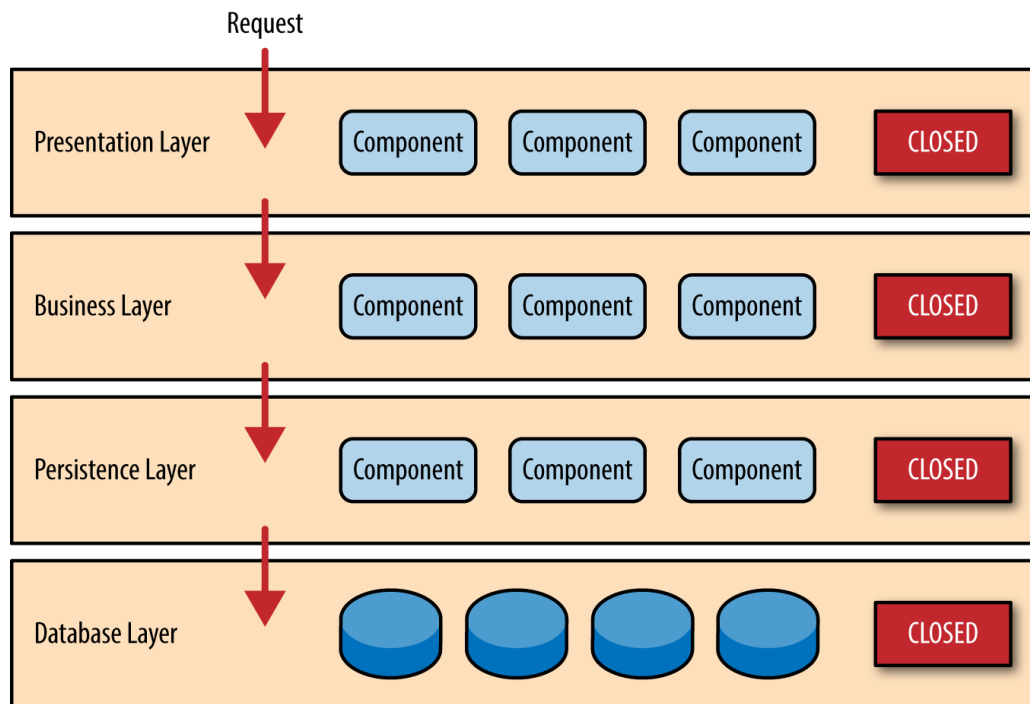


Рисунок 1.1 – Приклад розділення додатку на шари

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.

Схеми 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

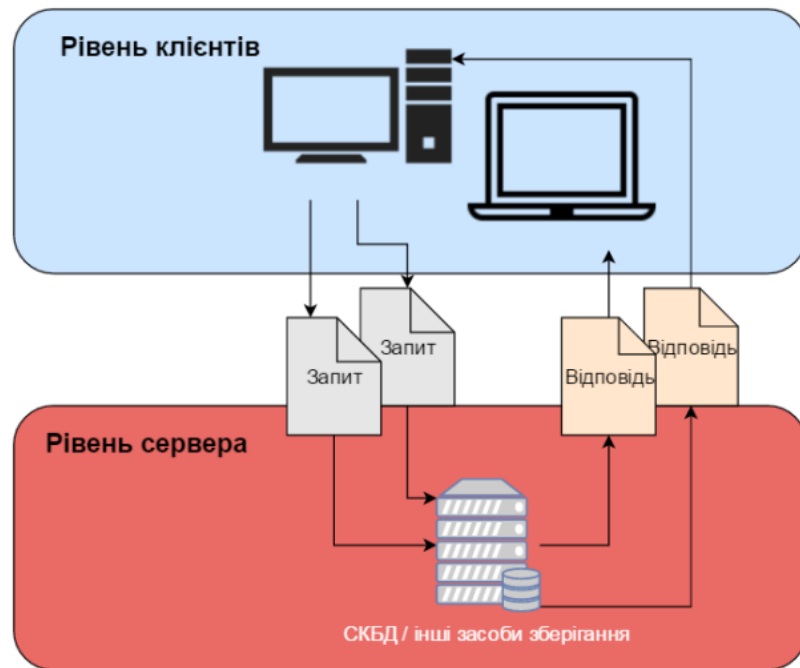


Рисунок 1.2 - Приклад двошарової архітектури

Головними перевагами такої архітектури є:

- простота системи, у порівнянні 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

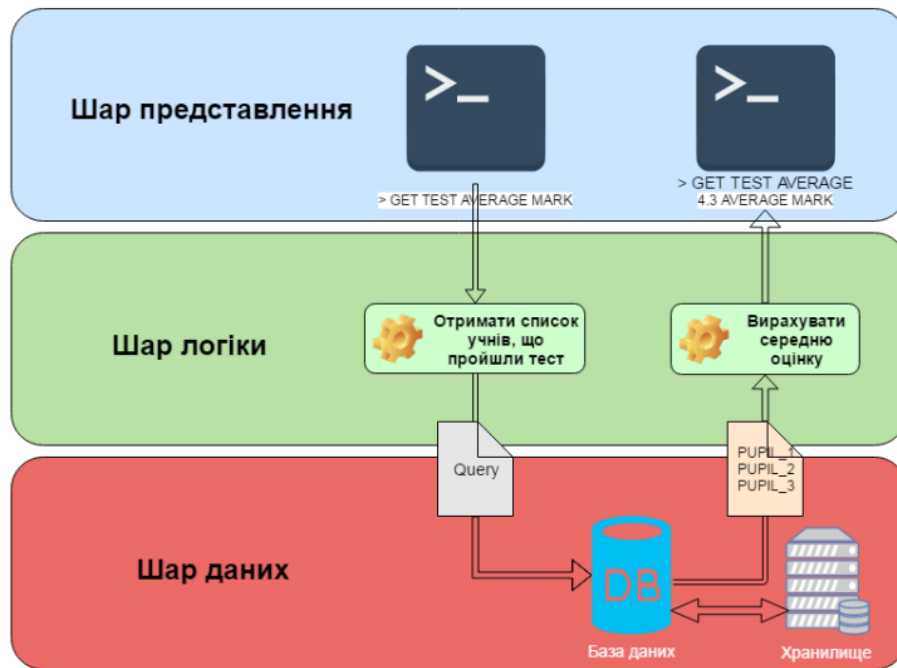


Рисунок 1.3 - Приклад трьохшарової архітектури

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

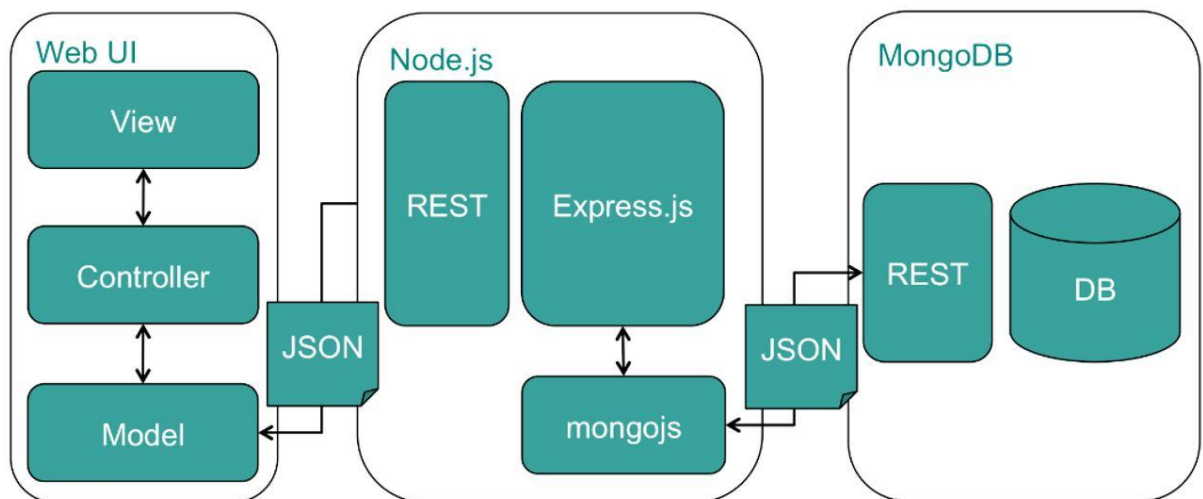


Рисунок 1.4 – Приклад багатошарової архітектури

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, розгортання 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.

На додачу до описаних шарів багатоварової архітектури 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 розширеного 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. [5]

Після аналізу досліджуваної предметної області було визначено, що найбільш ефективним рішенням для реалізації програмного застосування є

багатошарова архітектура. Це дає змогу створити зручний та масштабований сервіс, який зможе оптимально опрацьовувати великі обсяги інформації.

1.2 Основний алгоритм

Основна ідея автоматизованої системи полягає в заміні ручного збору та обробки статистичної інформації на її автоматизований збір та обробку в комп'ютерній системі. Для досягнення цієї мети, створюється відповідна інформаційна система, яка міститиме базу даних з інформацією про категорії статистичних даних, користувачів та статистичні дані.

Проектування бази даних починається з концептуального проектування, де визначаються всі об'єкти, що використовуються в базі даних, їх характеристики та зв'язки між ними. База даних розташовується в хмарі та автоматично створюється у разі її відсутності.

Після з'єднання з базою даних, програма отримує доступ до виконання різноманітних запитів в базу даних, таких як додавання, редагування, видалення та читання інформації. Також, система передбачає підтримку авторизації користувачів та обробку даних з форм.

У результаті роботи автоматизованої системи статистична інформація буде збиратися та зберігатися в базі даних, яка забезпечить швидкий та легкий доступ до неї, а також можливість проведення різноманітного аналізу.

2 ПОСТАНОВА ЗАВДАННЯ

2.1 Мета створення програми

Метою дипломної роботи є створення онлайн-сервісу, на основі багатошарової клієнт-серверної архітектури, який дозволить користувачам зберігати та аналізувати статистичні дані

2.2 Функції програми

Після проведення дослідження відповідної предметної області, необхідно розробити програму з різноманітними функціями, щоб забезпечити користувачам можливість ефективно використовувати її потенціал. Нижче наведено функції, які повина включати програма:

- авторизація за допомогою облікового запису Google для зручного входу в систему без необхідності створювати новий обліковий запис та запам'ятовування логіну та паролю;
- можливість створювати, редагувати, видаляти та змінювати порядок категорій;
- користувачі повинні мати змогу переглядати інформацію про категорії в зручному табличному форматі;
- можливість створювати, редагувати, видаляти та змінювати порядок груп категорій. Одна категорія може належати до декількох груп;
- можливість створювати, редагувати, видаляти та змінювати порядок підкатегорій. Підкатегорії повинні значно полегшити статистичний облік даних користувачам;
- можливість створювати, редагувати та видаляти записи статистики в системі на основі категорій. Ця функція дозволяє зберігати важливі дані та інформацію;
- можливість перегляду активних сеансів користувача та їх завершення в разі необхідності;

- візуальне відображення статистичних даних у вигляді графіка з різноманітними параметрами групування та фільтрації;
- можливість переглядати записи статистики у вигляді таблиці з фільтраціями та сортуваннями за різними критеріями;
- можливість безпечного виходу з облікового запису без втрати даних;
- забезпечення безпеки даних. Забезпечення цілісності та безпеки даних є найважливішою функцією, яка гарантує, що інформація користувачів буде захищена від несанкціонованого доступу.

2.3 Вимоги до проектованої системи

Основні вимоги до проектованої системи включають:

- забезпечення чіткого та зрозумілого інтерфейсу взаємодії з користувачем, що містить всі необхідні елементи та легкий у використанні, а також адаптивний до різних розмірів екранів;
- забезпечення зрозумілості використання програми, включаючи наявність текстових описань дій та підказок;
- збереження даних, які завантажує користувач на сервері, та забезпечення їх доступності у будь-який момент часу;
- забезпечення правильної роботи всіх функцій програми та можливості скасування змін;
- забезпечення надійності та конфіденційності персональних даних користувача, які використовує програма;

Додаткові вимоги до проектованої системи включають:

- забезпечення швидкої та ефективної роботи системи;
- сумісність з різними операційними системами та браузерами;
- забезпечення можливості редагування та видалення даних користувача;
- забезпечення можливості інтеграції з іншими програмами або сервісами;

- забезпечення можливості масштабування та розвитку системи в майбутньому;

2.4 Вимоги до надійності

До основних вимог до надійності та безпеки програми відносяться наступні:

- забезпечення конфіденційності та цілісності персональних даних користувача. Це означає, що будь-яка інформація, яку користувач надає в рамках програми, повинна бути захищена від несанкціонованого доступу і використання третіми особами. Також необхідно забезпечити захист від можливих атак хакерів та зловмисників;

- повідомлення користувача про будь-які помилки або проблеми, що виникають при завантаженні або роботі програми. Користувач повинен бути повідомлений про будь-які виникненні помилки та отримувати достатньо інформації для їх виправлення;

- забезпечення індивідуального доступу до даних. Кожен користувач повинен мати можливість зайти в програму лише за своїм акаунтом і мати доступ тільки до своїх власних даних. Таким чином, забезпечується захист персональних даних від доступу третіх осіб;

- програма повинна чітко інтерпретувати та зберігати інформацію. Всі дані, введені користувачем, повинні бути збережені на сервері програми та бути доступними для перегляду та редагування користувачем в будь-який момент часу. Програма повинна чітко інтерпретувати цю інформацію та відображати її у зрозумілому форматі для користувача. Також необхідно забезпечити надійне зберігання даних на сервері та їх резервне копіювання, щоб у разі виникнення проблем з сервером, дані користувача були збережені в безпечному місці.

2.5 Умови роботи програми

Для коректної роботи клієнтської частини програми необхідно мати браузер, який має мінімальну версію не нижче 2020 року. Найбільш підходящими для використання є такі браузери, як Google Chrome, Safari, Mozilla Firefox, Opera та Microsoft Edge.

Рекомендується мати останню версію браузера, оскільки це забезпечує оптимальну швидкість роботи та захист від можливих вразливостей. Також рекомендується використовувати встановлені оновлення та плагіни для браузера для запобігання можливих проблем з безпекою.

Як можна бачити, згідно статистики популярності різних браузерів “Desktop Browser Market Share Worldwide Mar 2022 - Mar 2023”, від statcounter, яка зображена на рисунку 1.5 клієнтська частина програми має коректно працювати у всіх сучасних та найбільш використовуваних браузерах. [6]

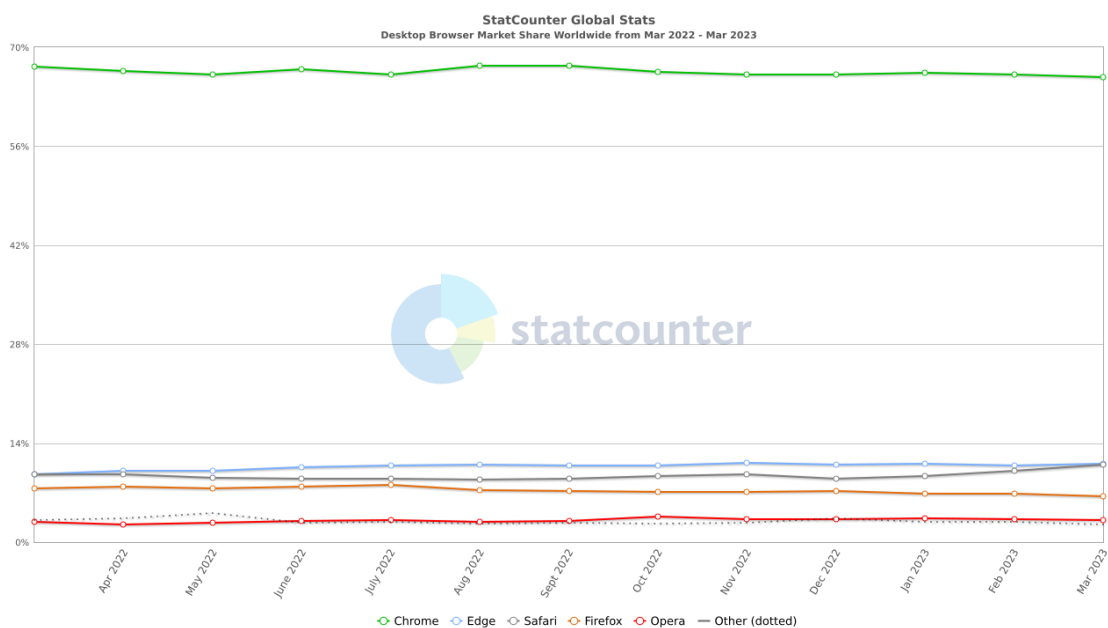


Рисунок 1.5 – Статистика популярності різних браузерів в світі

Серверна частина програмного забезпечення працює на NodeJS - це засіб виконання JavaScript, який дозволяє розробникам створювати серверні додатки. Різні версії NodeJS мають різний рівень підтримки нових функцій та можуть мати

відмінності в швидкості та стабільності роботи. Для того, щоб забезпечити правильну роботу сервісу, рекомендується використовувати одну з наступних LTS версій NodeJS: 18.15.0, 16.20.0 або 14.21.3.

Варто відзначити, що NodeJS на сьогодні є найпопулярнішою платформою для розробки клієнт-серверних додатків, оскільки вона дозволяє розробникам писати серверний код на JavaScript, що дозволяє швидко та ефективно створювати високоякісні додатки.

На рисунку 1.6 зображена статистика найбільш використовуваних веб-фреймворків та бібліотек за 2022 рік. Як можна бачити, NodeJS займає перше місце [7]

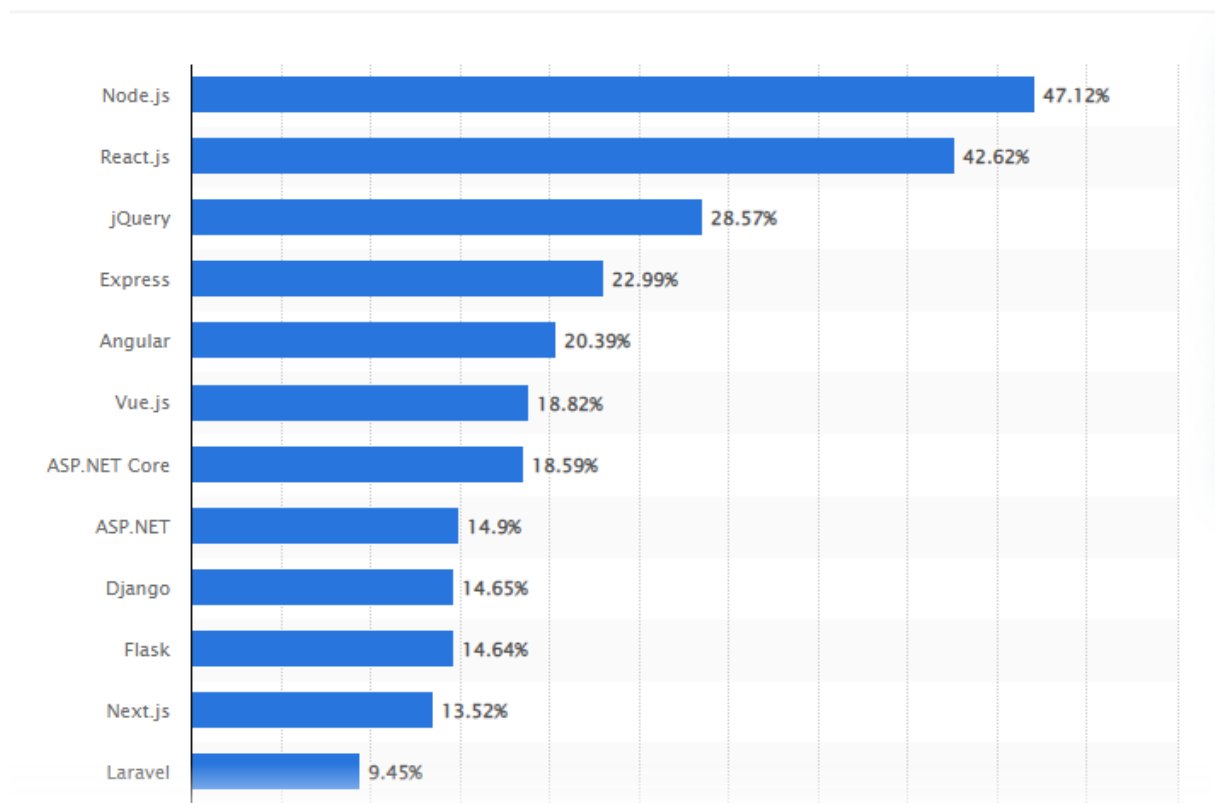


Рисунок 1.6 – Статистика найбільш використовуваних веб-фреймворків та бібліотек за 2022 рік

2.6 Умови розповсюдження програми

Даний програмний продукт є безкоштовним і доступним за допомогою мережі Інтернет. Щоб мати можливість користуватися програмою, користувачеві необхідно увести в адресний рядок свого браузера наступну адресу: "<https://counter-ltlaitoff.vercel.app>". Ця адреса є посиланням на веб-сторінку, на якій розміщений даний програмний продукт. Веб-сторінка містить інформацію про програму та надає можливість користувачам скористатися нею безкоштовно. Варто відзначити, що таке розповсюдження дозволяє забезпечити швидкий та зручний доступ до програмного продукту для широкого кола користувачів з усього світу.

3 ПРОГРАМУВАННЯ

3.1 Обґрунтування вибору середовища розробки системи

В якості середовища розробки сервісу був обраний Visual Studio Code — це легкий, але потужний редактор вихідного коду, який може працювати навіть в браузері та доступний для Windows, macOS і Linux. Поставляється з вбудованою підтримкою JavaScript, TypeScript і Node.js і має багату екосистему розширень для інших мов і середовищ виконання (таких як C++, C#, Java, Python, PHP, Go, .NET). Вигляд основного вікна Visual Studio Code зображений на рисунку 3.1 [8]

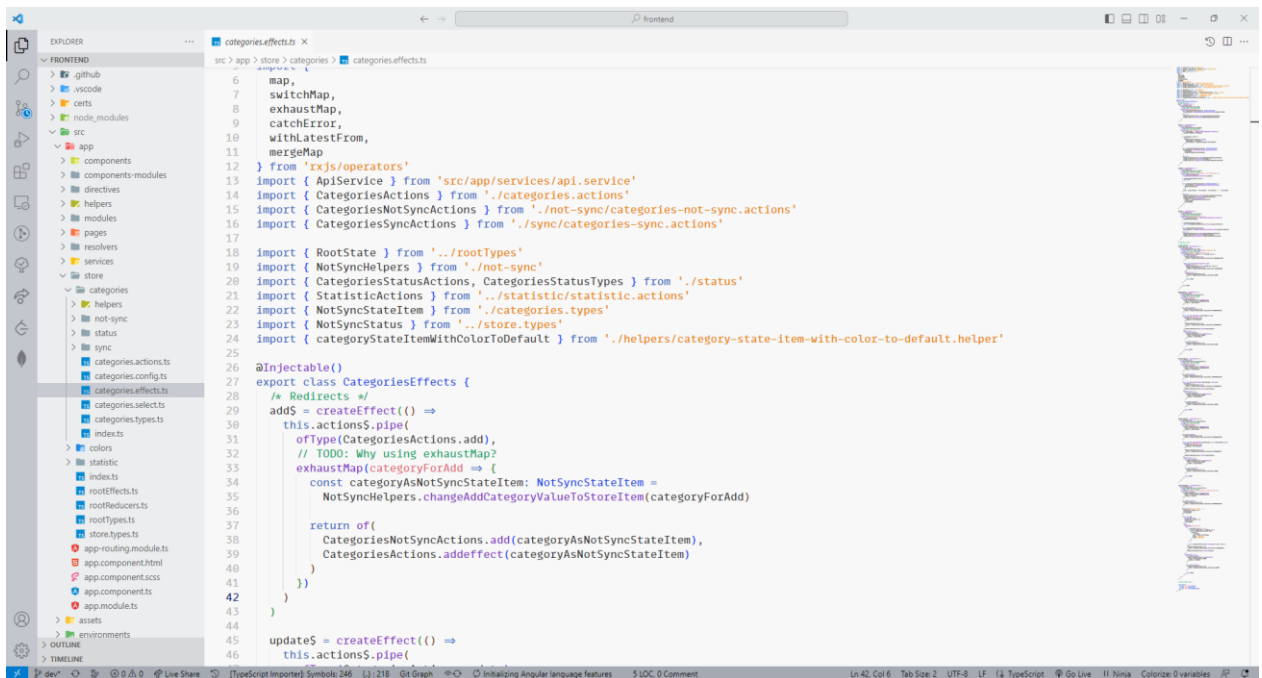


Рисунок 3.1 – Вигляд основного вікна

Він підтримує ряд мов програмування, зокрема мову програмування TypeScript, яка була використана для написання сервісу, підсвічування синтаксису, IntelliSense, рефакторинг та налагодження програми, навігацію по коду, підтримку Git та інші можливості для розробки різноманітних додатків

Visual Studio Code є оптимальним рішенням для розробки цільного додатку. Він підтримує велику кількість мов, що стане в нагоді при розробці проектів, в яких використані різні рішення щодо реалізації їх компонентів.

Також в цьому середовищі розробки є вбудована функція відкриття і перегляду не тільки одного файлу, а й цілої папки проекту: її вміст буде відображатись у лівій частині інтерфейсу. Це робить зручним швидкий доступ до різних файлів проекту і налагодження їх зв'язку між собою, а разом з підтримкою багатьох рішень розробки дозволяє швидко перемикаати увагу з однієї частини проекту на іншу, не витрачаючи часу на відкриття інших програм.

Великою перевагою VS Code є присутній у ньому вбудований відладчик, меню якого показано на рисунку 3.2, програм майже для будь-якої обраної мови програмування. Є можливість запустити файл на виконання не переходячи в інші програми, що прискорює знаходження і виправлення помилок.

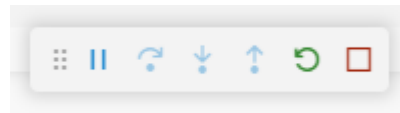


Рисунок 3.2 – Демонстрація панелі для роботи з вбудованим відладчиком Visual Studio Code

Також реалізовані функції навігації по коду, доповнення типових конструкцій і контекстної підказки(IntelliSense), які суттєво прискорюють роботу над проектом. Контекстні підказки показані на рисунку 3.3

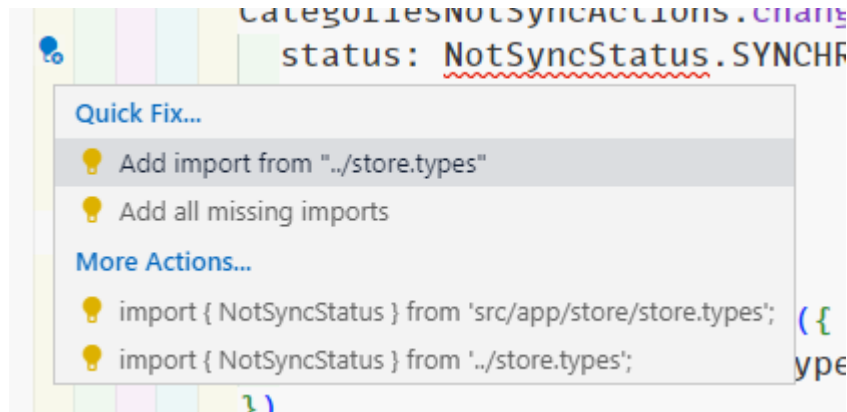


Рисунок 3.3 – Демонстрація контекстних підказок

Основною перевагою Visual Studio Code є реалізована зручна вбудована система управління надлаштуваннями і розширеннями: їх можна завантажувати і встановлювати прямо у редакторі на вкладинці Extensions. До розширень належать, наприклад, пакети підтримки мов, бібліотеки або відладчики до них, додатки для підсвітки і форматування коду та інші надлаштування, що розширюють його функціонал або підвищують комфортність роботи у редакторі. На рисунку 3.4 продемонстрована робота розширення «GitLens»

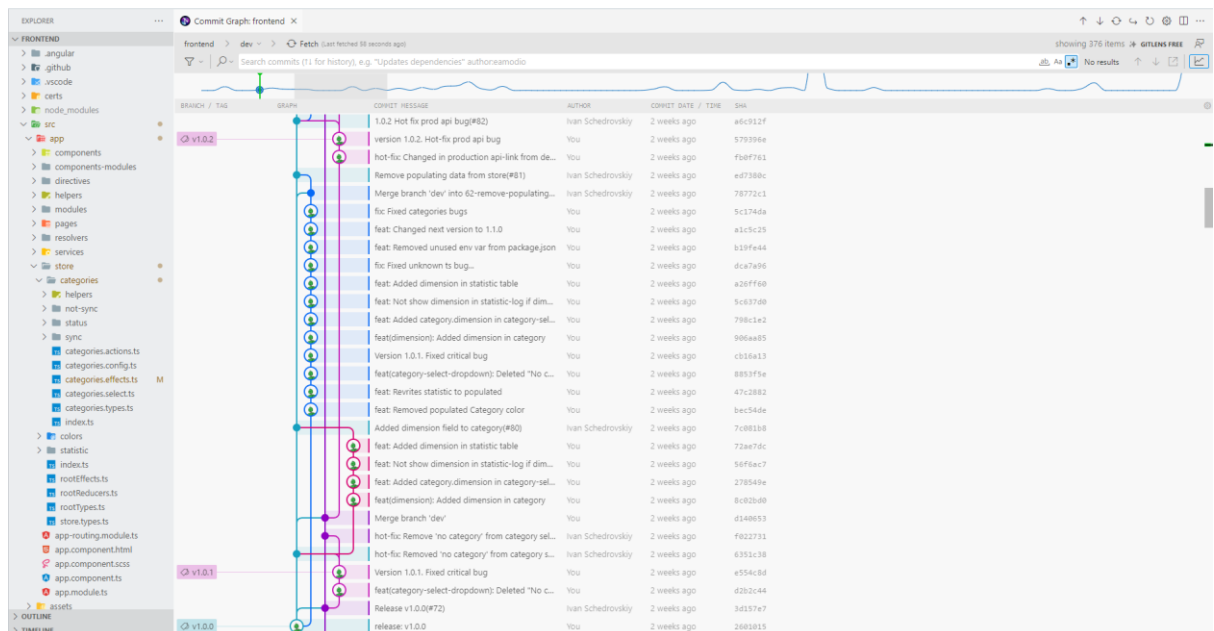


Рисунок 3.4 – Демонстрація роботи розширення «GitLens»

Важливою особливістю VS Code є простий і зрозумілий інтерфейс. Всі елементи розбиті на групи, їх небагато, що робить роботу в редакторі зручною і легко зрозумілою навіть для новачка. Також інтерфейс можна повністю налаштувати: від зовнішнього вигляду і теми до положення елементів управління. Користувач може змінити вікно програми для максимізації власного комфорту, або спеціально під розробку конкретного додатку

В якості систему контролю версіями був обраний Git.

Система контролю версій - це система, що записує зміни у файл або набір файлів протягом деякого часу, так щоб розробник зміг повернутися до певної версії пізніше. [9]

У 2005 році відносини між спільнотою розробників ядра Linux і комерційною компанією, що розробила BitKeeper почали псуватись, і безкоштовне використання продуктом було скасовано. Це підштовхнуло розробників Linux (і зокрема Лінуса Торвальдса, автора Linux) розробити власну систему, ґрунтуючись на деяких з уроків, які вони дізналися під час використання BitKeeper. [10]

Як видно з графіку зображеного на рисунку 3.5 популярності різних систем контролю версій 2022 року серед професійних розробників від StackOverFlow на даний момент Git є найбільш популярною системою контролю версій. [11]



Рисунок 3.5 – Графік популярності різних систем контролю версій 2022 року серед професійних розробників від StackOverflow

Для зручної побудови та тестування API, зокрема зі стандартом REST, було обрано Postman.

API (Application Programming Interface) забезпечує взаємодію між двома системами. API дозволяє надсилати інформацію безпосередньо з однієї програми до іншої, обминаючи інтерфейс взаємодії з користувачем. API приймає запит, передає інформацію системі, обробляє її та повертає відповідь.

REST (Representational State Transfer) – стандарт архітектури взаємодії додатків і сайтів, що використовує протокол HTTP. Особливість REST в тому, що сервер не запам'ятовує стан користувача між запитами. Іншими словами, ідентифікація користувача і всі параметри виконання операції передаються в кожному запиті.

Для передачі запитів і відповідей під час роботи з API використовується протокол HTTP

HTTP (Hypertext Transfer Protocol) – це протокол передачі гіпертексту, який дозволяє клієнту та серверу спілкуватися по мережі за допомогою запиту/відповіді. Це протокол рівня додатка, який покладається на TCP/IP для своїх послуг.

HTTPS – це розширення протоколу передачі гіпертексту(HTTP). Він використовується для безпечного спілкування через комп'ютерну мережу. [12]

Postman - це платформа для розробки та використання API, яка дозволяє спростити кожен етап процесу створення та вдосконалення API, включаючи тестування та документування. Завдяки Postman розробники можуть створювати API швидше і ефективніше.

Тестування в Postman є надзвичайно зручним завдяки багатьом корисним можливостям, які ця платформа надає. Розробники можуть створювати тести, які виконують запити до програми і перевіряють, чи все працює належним чином або, навпаки, не працює.

Postman надає зручний інтерфейс для створення тестів. Розробник може написати скрипт, який виконує запити до веб-сервісу, передаючи йому різні дані і

параметри. Скрипт також може перевіряти відповіді сервера, переконуючись, що отримані дані відповідають очікуваному результату.

Однією з важливих можливостей Postman є можливість автоматичного тестування. Розробники можуть створювати набори тестів, які виконуються автоматично при кожній зміні в коді або перед релізом програми. Це дозволяє виявити проблеми та помилки швидше і забезпечує стабільність програми.

Після виконання тестів, Postman надає детальні звіти про результати. Розробники можуть переглянути, які тести пройшли успішно, а які не пройшли і потребують виправлень. Це допомагає виявити проблемні місця в програмі та швидко їх виправити.

Крім того, платформа Postman забезпечує візуалізацію запитів і відповідей, що полегшує налагодження API та дозволяє зменшити час на коригування помилок. [13]

Вигляд головного вікна Postman зображено на рисунку 3.6

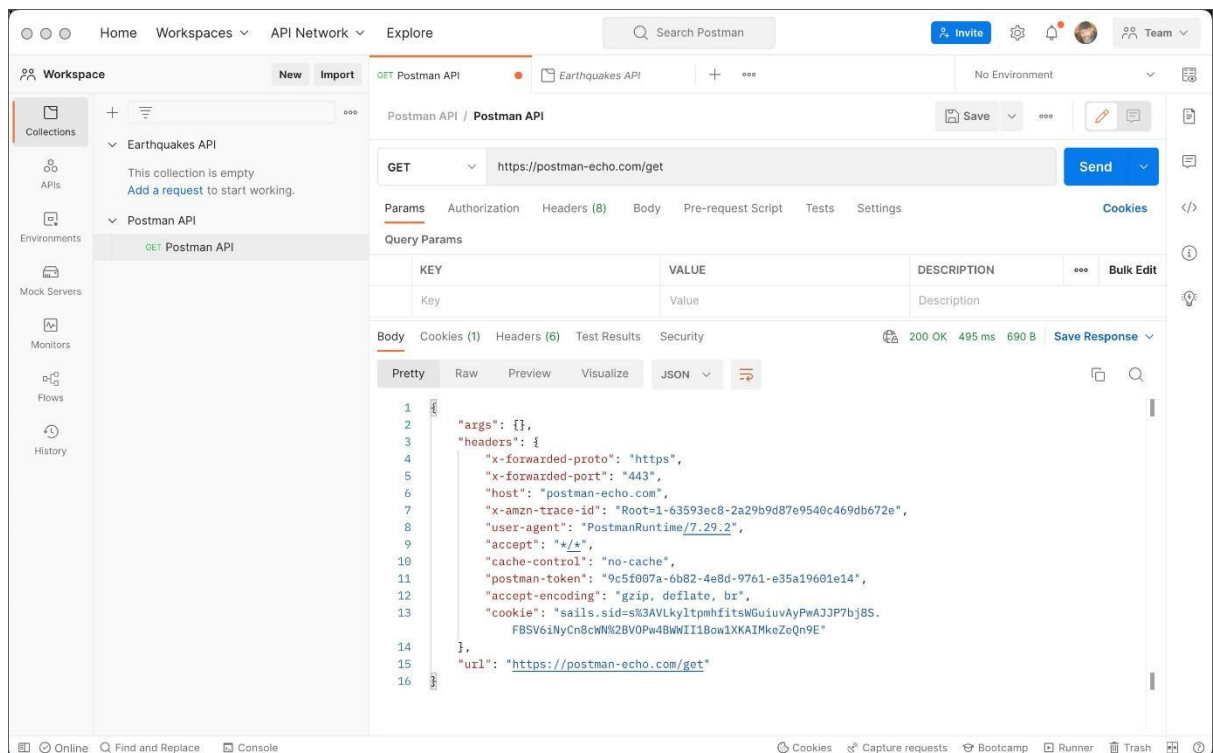


Рисунок 3.6 – Вигляд головного вікна Postman

Swagger UI дозволяє візуалізувати та взаємодіяти з ресурсами API, не маючи жодної реалізації логіки. Вона автоматично генерується з OpenAPI (раніше відомого як Swagger) специфікації, а візуальна документація спрощує реалізацію на боці сервера та використання на боці клієнта.

Swagger UI в дипломній роботі використовується для швидкої розробки документації API для більш зручного використання серверної частини при розробці клієнтської та може використовуватись іншими розробниками при роботі з API в майбутньому [14]

3.2 Обґрунтування вибору середовища функціонування системи

Клієнтська частина сервісу повинна функціонувати або ж в браузері користувача, або в вигляді PWA застосунку

Браузер – програмне забезпечення яке переносить користувача до будь-якого куточку Інтернету і дає змогу переглядати текст, зображення та відео де б користувач не знаходився.

Браузер переносить вас до будь-якого закуточка Інтернету. Він бере інформацію з інших частин Інтернету та показує її на вашому комп'ютері чи мобільному пристрої. Інформація передається за допомогою протоколу передачі гіпертексту (Hypertext Transfer Protocol), який визначає, як передаються текст, зображення та відео в Інтернеті. Ця інформація повинна надсилатися та показуватися у сумісному форматі, щоб користувачі будь-якого браузера, у будь-якому куточку світу, могли побачити цю інформацію. [15]

Progressive Web Applications (PWA), яка була анонсована Google у 2015 році. PWA — це сучасні вебсайти, які наділені характеристиками нативних додатків: можливістю запуску з робочого столу девайсу із доступом до його вбудованих функцій, спроможністю роботи offline, наявністю push-сповіщень тощо.

Виявлено, що особливі переваги від використання PWA можуть отримати компанії у сфері e-commerce й інформаційного бізнесу. Зручність використання таких додатків, високий рівень UI/UX, їх швидкодія та невибагливість до ресурсів, можливість роботи без підключення до інтернету сприяють розширенню мобільної присутності компаній у вебпросторі, зростанню конверсії та збільшенню доходів, які надходять з інтернет-каналів. Важливою перевагою є невисока вартість розробки та супроводу PWA у порівнянні з їх нативними аналогами, що пояснюється кросплатформеністю готового продукту

Створення прогресивного вебдодатку у порівнянні з нативним обходиться компаніям у середньому в 3-4 рази дешевше, у деяких випадках у 10-15 разів. Це пояснюється необхідністю розробки окремих додатків не тільки під різні платформи, а й версії операційних систем. Так вартість нативного додатку для App або Play Store починається від 10 тис. дол. США.[16]

PWA дозволяє значно зменшити розмір застосунку на мобільних додатках. На рисунку 3.7 приведено графічне порівняння розмірів застосунку “Pinterest” використовуючи нативні застосунки та PWA.[17]

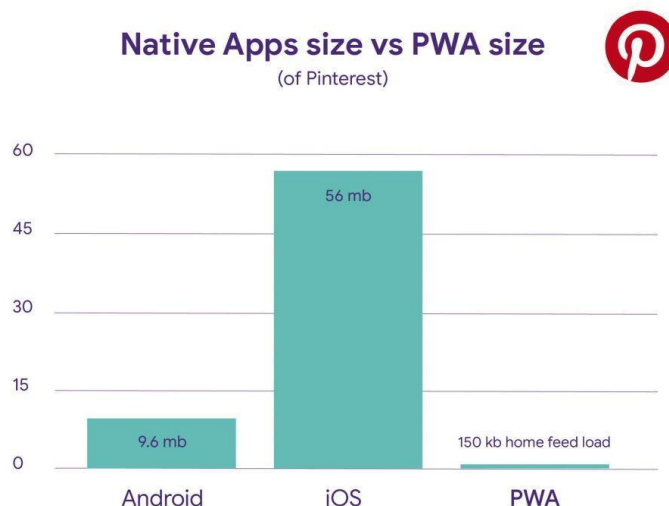


Рисунок 3.7 – Графічне порівняння розмірів застосунку “Pinterest” використовуючи нативні застосунки та PWA

Для коректної роботи клієнтської частини програми необхідно мати браузер, який має мінімальну версію не нижче 2020 року. Найбільш підходящими для використання є такі браузери, як Google Chrome, Safari, Mozilla Firefox, Opera та Microsoft Edge.

Рекомендується мати останню версію браузера, оскільки це забезпечує оптимальну швидкість роботи та захист від можливих вразливостей. Також рекомендується використовувати встановлені оновлення та плагіни для браузера для запобігання можливих проблем з безпекою.

Як можна бачити, згідно статистики популярності різних браузерів “Desktop Browser Market Share Worldwide Mar 2022 - Mar 2023”, від statcounter, яка зображена на рисунку 3.8 клієнтська частина програми має коректно працювати у всіх сучасних та найбільш використовуваних браузерах. [6]

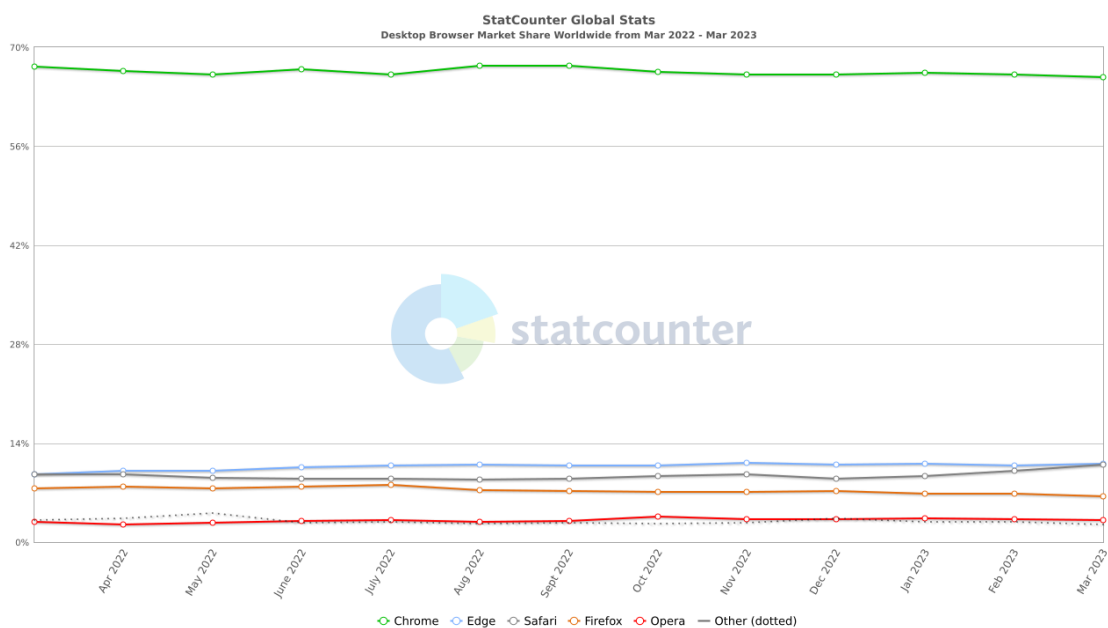


Рисунок 3.8 – Статистика популярності різних браузерів в світі

Серверна частина програмного забезпечення працює на NodeJS - це засіб виконання JavaScript, який дозволяє розробникам створювати серверні додатки.

Різні версії NodeJS мають різний рівень підтримки нових функцій та можуть мати відмінності в швидкості та стабільності роботи. Для того, щоб забезпечити правильну роботу сервісу, рекомендується використовувати одну з наступних LTS версій NodeJS: 18.15.0, 16.20.0 або 14.21.3.

Варто відзначити, що NodeJS на сьогодні є найпопулярнішою платформою для розробки клієнт-серверних додатків, оскільки вона дозволяє розробникам писати серверний код на JavaScript, що дозволяє швидко та ефективно створювати високоякісні додатки.

На рисунку 3.9 зображена статистика найбільш використовуваних веб-фреймворків та бібліотек за 2022 рік. Як можна бачити, NodeJS займає перше місце [7]

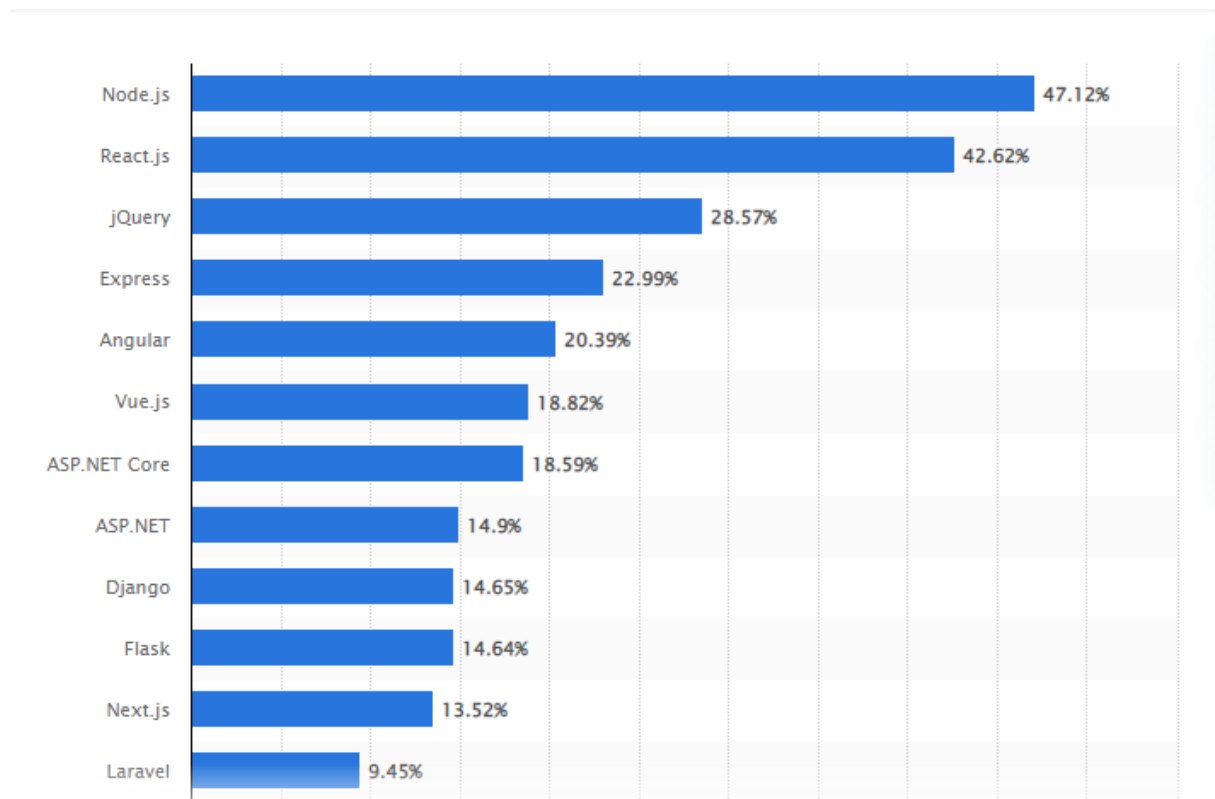


Рисунок 3.9 – Статистика найбільш використовуваних веб-фреймворків та бібліотек за 2022 рік

3.3 Основні рішення щодо реалізації компонентів системи

3.3.1 Використовувані моделі даних

Для розробки даного сервісу використовується документо-орієнтована модель даних. На рисунку 3.10 представлений приклад документо-орієнтованої моделі

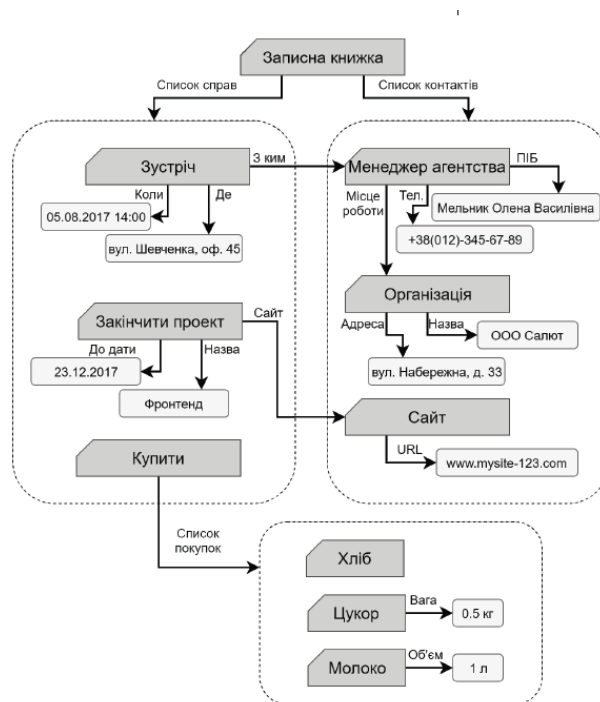


Рисунок 3.10 - Приклад документо-орієнтованої моделі даних

Перевагами документо-орієнтованої моделі бази даних є:

- семантичне наповнення. Дані у документо-орієнтованій базі даних не можуть бути описані за допомогою єдиної схеми. Замість цього, дані описують самі себе, тобто кожен елемент даних має особисту семантичну схему;
- структурна незалежність та незалежність даних. Характерною особливістю слабоструктурованих даних є те, що описова інформація, яка зазвичай виділяється в окрему схему, присутня в самих даних. Тому дані в документо-орієнтованій моделі не залежать від будь-яких зовнішніх структур та схем;

- відповідність стандартам опису даних. Наявність загальних стандартів для представлення даних, таких як XML, JSON, YAML;
- простота проектування, реалізації, керування та використання;
- висока масштабованість;
- ключові переваги NoSQL баз в розподілених системах полягають в процедурах шаринга і реплікації.

Недоліками документо-орієнтованої моделі даних:

- відсутність транзакцій. Відсутність єдиної схеми та специфіка області застосування документо-орієнтованих баз даних поки що не дають змоги реалізувати повноцінний механізм транзакцій;
- відсутність стандартизованої мови маніпулювання даними;
- недостатня цілісність даних. Задача контролю цілісності даних ніяк не регулюється системою контролю баз даних та повинна бути реалізована у коді додатків, що працюють з цією базою даних. [18]

В роботі використовується MongoDB - крос-платформна, потужна, гнучка, документно-орієнтована база даних, що легко масштабується.

MongoDB складається з баз даних, які зберігають в собі колекції.

Колекція – іменована безліч об'єктів, при цьому один об'єкт належить лише одній колекції.

Об'єкт – сукупність властивостей, включаючи унікальний ідентифікатор `_id`.

Властивість – сукупність назви і відповідного йому типу і значення.

Типи властивостей – `string`, `number`, `float`, `array`, `object`, `buffer`, `date`, `boolean`, `null`, `objectId`.

Підтримуються операції вибірки (`count`, `group`, `MapReduce`), вставки, зміни та видалення.

Зв'язків між об'єктами немає, об'єкти можуть лише зберігати інші об'єкти у властивостях.

Підтримуються як унікальні, так і композитні індекси. Індеси можна накладати на властивості вкладених об'єктів. Швидкість MongoDB гарно себе показує при виконанні операції вставки даних, роблячи їх дуже швидко. [19]

Формат зберігання і формат передачі об'єктів по мережі один і той же, так що для вибірки якогось об'єкта треба всього лише знайти його позицію за індексом і повернути клієнту шматок файлу певної довжини.

В якості унікального ідентифікатора використовується 12-байтне унікальне число, що генерується на клієнті.

По-перше немає проблеми з синхронізацією реплік, тобто можна незалежно робити вставки на дві різні машини, і конфлікту не виникне. По-друге, не буде нісенітниць з переповненням цілого числа, ну і після перевтілення бази даних, пошукачі не будуть адресувати на нові статті за старими посиланнях.

В процесі проектування даного програмного продукту для побудови об'єктно-орієнтованої моделі було виділено наступні об'єктні множини: “User”, “Statistic”, “Category”, “Color”, “CategoryGroup”

Аналіз визначених об'єктів і атрибутів дозволяє виділити сутності бази даних і побудувати її логічну схему.

Спроектowana документo-орієнтована модель зображена на рисунку 3.11

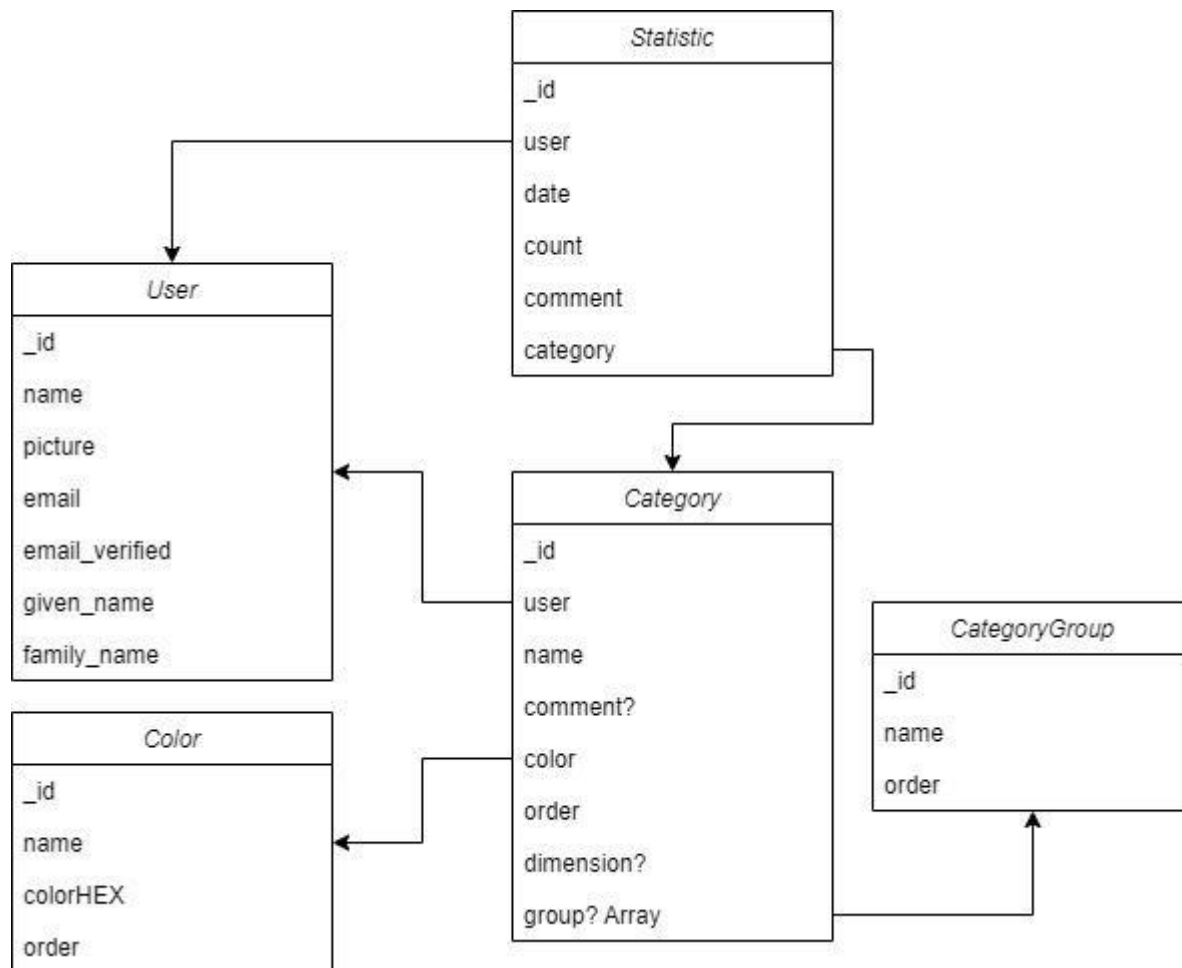


Рисунок 3.11 - Спроектowana модель даних

Документ «User» (_id, name, picture, email, email_verified, given_name, family_name) зберігає інформацію про користувачів. Опис документа «User» представлений в таблиці 3.1, схема документа представлена на рисунку 3.12

Таблиця 3.1 – Опис документа «User»

Назва ключа	Тип значення	Опис
_id	ObjectID	Унікальне поле
name	string	Ім'я та прізвище користувача
picture	string	Аватар
email	string	Електронна пошта
email_verified	boolean	Чи підтверджена пошта
given_name	string	Ім'я користувача

Продовження таблиці 3.1

Назва ключа	Тип значення	Опис
family_name	string	Прізвисько користувача

```
@Schema()
export class User {
  @Prop({ required: true })
  name: string

  @Prop({ required: true })
  picture: string

  @Prop({ required: true })
  email: string

  @Prop({ required: true })
  email_verified: boolean

  @Prop({ required: true })
  given_name: string

  @Prop({ required: true })
  family_name: string
}
```

Рисунок 3.12 – Схема документа «User»

Документ «Statistic» (_id, user, date, count, comment, category) зберігає інформацію про користувачів. Опис документа «Statistic» представлений в таблиці 3.2, схема документа представлена на рисунку 3.13

Таблиця 3.2 – Опис документа «Statistic»

Назва ключа	Тип значення	Опис
_id	ObjectID	Унікальне поле
user	ObjectID	Користувач . Посилання на User
date	Date	Дата запису

Продовження таблиці 3.2

count	number	Кількість
comment	string	Коментар
category	ObjectID	Категорія. Посилання на Category

```
@Schema()
export class Statistic {
  @Prop({ type: mongoose.Schema.Types.ObjectId,
ref: 'User', required: true })
  user: User

  @Prop({ required: true })
  date: Date

  @Prop({ required: true })
  count: number

  @Prop()
  comment: string

  @Prop({
    type: mongoose.Schema.Types.ObjectId,
    ref: 'Category',
    required: true
  })
  category: Category
}
```

Рисунок 3.13 – Схема документа «Statistic»

Документ «Category» (_id, user, name, comment, color, order, dimension, group) зберігає інформацію про користувачів. Опис документа «Category» представлений в таблиці 3.3, схема документа представлена на рисунку 3.14

Таблиця 3.3 – Опис документа «Category»

Назва ключа	Тип значення	Опис
_id	ObjectID	Унікальне поле
user	ObjectID	Користувач. Посилання на User
name	string	Назва
comment	string	Кількість
color	ObjectID	Колір. Посилання на Color
order	number	Порядок категорії
dimension	string	Одиниці виміру
group	ObjectID	Група категорії. Посилання на CategoryGroup

```

@Schema()
export class Category {
  @Prop({ type: mongoose.Schema.Types.ObjectId,
ref: 'User', required: true })
  user: User

  @Prop({ required: true })
  name: string

  @Prop()
  comment?: string

  @Prop({ type: mongoose.Schema.Types.ObjectId,
ref: 'Color', required: true })
  color: Color

  @Prop({ required: true })
  order: number

  @Prop()
  dimension?: string

  @Prop({ type:
[mongoose.Schema.Types.ObjectId], ref: 'Group' })
  group?: [Group]
}

```

Рисунок 3.14 – Схема документа «Category»

Документ «Color» (_id, name, colorHEX, order) зберігає інформацію про користувачів. Опис документа «Color» представлений в таблиці 3.4, схема документа представлена на рисунку 3.15

Таблиця 3.4 – Опис документа «Color»

Назва ключа	Тип значення	Опис
_id	ObjectID	Унікальне поле
name	string	Назва кольору
colorHEX	string	Колір в HEX форматі
order	number	Порядок кольору

```
@Schema()  
export class Color {  
  @Prop({ required: true })  
  name: ColorsNames  
  
  @Prop({ required: true })  
  colorHEX: string  
  
  @Prop({ required: true })  
  order: number  
}
```

Рисунок 3.15 – Схема документа «Color»

Документ «CategoryGroup» (_id, name, order) зберігає інформацію про користувачів. Опис документа «CategoryGroup» представлений в таблиці 3.5, схема документа представлена на рисунку 3.16

Таблиця 3.5 – Опис документа «CategoryGroup»

Назва ключа	Тип значення	Опис
_id	ObjectID	Унікальне поле
name	string	Назва групи категорій
order	number	Порядок групи категорій

```
@Schema()
export class Group {
  @Prop({ required: true })
  name: string

  @Prop({ required: true })
  order: number
}
```

Рисунок 3.16 – Схема документа «CategoryGroup»

3.3.2 Структурна схема програми

Загальна схематична система, зображена на рисунку 3.17, складається з:

- MongoDB – база даних;
- сервер – програма, яка приймає та відправляє запити від клієнта, працює з базою даних. Операції між сервером та базою, а також сервером та клієнтом відбуваються за допомогою Rest API;
- клієнт – користувач, який взаємодіючи з сервісом відправляє та отримує запити від сервера.

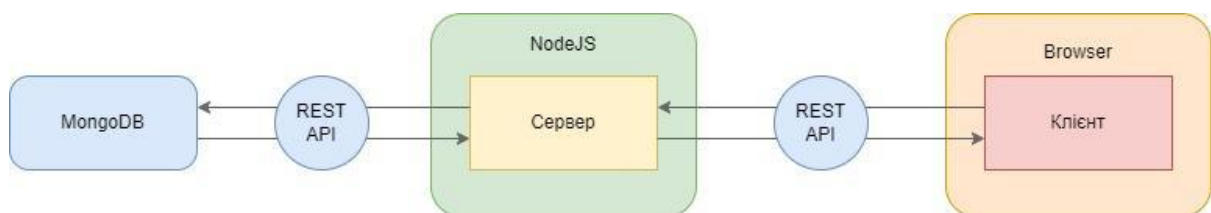


Рисунок 3.17 – Загальне схематичне зображення роботи сервісу

Повна схематична система, зображена на рисунку 3.18 складається з:

- MongoDB – база даних;
- Mongoose – пакет для роботи з базою даних;
- NestJS – фреймворк для побудови API;
- Angular – фреймворк для клієнтської частини.

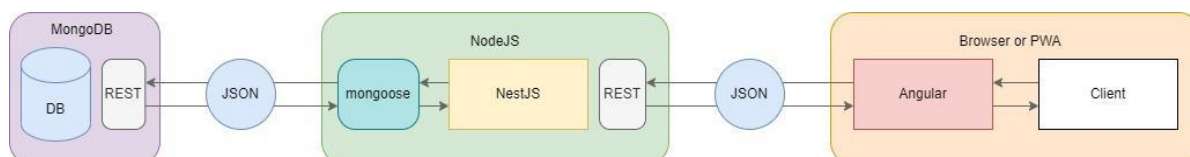


Рисунок 3.18 – Повне схематичне зображення роботи сервісу

3.3.3 Розробка модулів системи

Основною та єдиною мовою програмування доступною в браузері є JavaScript, тому її використання для розробки сервісу є логічним і необхідним

JavaScript – легка, інтерпретована, об'єктно-орієнтована мова з функціями першого класу, а також найвідоміша скриптова мова для веб-сторінок, але також використовується в багатьох не браузерних оточеннях.

Інтерпретована означає, що сирцевий код програми не перетворюється повністю у машинний код для виконання, як у компільованих, а виконується рядок за рядком з допомогою спеціальної програми інтерпретатора

Доказом того, що JavaScript - інтерпретована мова програмування є дещо різна поведінка одного й того самого коду в різних оточеннях

JavaScript є об'єктно-орієнтованою мовою через те, що в основі мови лежить поняття об'єкта – певної сутності, яка об'єднує в собі поля(дані) і методи(виконувані об'єктом дії)

JavaScript має функції першого класу, бо розглядає функції як об'єкти першого класу, тобто сутність, яка може бути побудована під час виконання

програми, передаватись як параметр, повертатись з підпрограми, або присвоюватись змінній. Це означає, що JavaScript підтримує передачу функції як аргументів інших функцій, повернення їх як результат інших функцій, присвоювання їх змінним або збереження в структурах даних

JavaScript може виконуватись як за допомогою браузерного оточення або, наприклад, NodeJS. NodeJS та більшість браузерів на даний момент використовують в якості рушія V8 який компілює JavaScript код безпосередньо у власний машинний код, минаючи стадію проміжного байт-коду, а також ефективно керує пам'яттю.

JavaScript - прототипно-орієнтована мультипарадигменна мова сценаріїв.

Прототипно-орієнтованою, бо в цій мові відсутнє поняття класу, а повторне використання (успадкування) проводиться шляхом клонування наявного примірника об'єкта – прототипу.

Мультипарадигменна – бо підтримує декілька підходів або стилів програмування: динамічний, об'єктно-орієнтований, імперативний та функціональний стилі.

JavaScript може функціонувати і як процедурна, і як об'єктно-орієнтована мова. Об'єкти можна створювати програмно під час виконання, шляхом приєднання методів і властивостей або порожніх об'єктів під час виконання, на відміну від синтаксичних визначень класів у мовах, що компілюються, таких як C++ або Java. Після того, як об'єкт був створений, він може бути використаний як план (або прототип) для створення схожих об'єктів.

Динамічні можливості JavaScript включають створення об'єктів під час виконання, змінну кількість параметрів, динамічне створення скриптів (за допомогою eval), перебір об'єктів (за допомогою for ... in), відновлення вихідного коду (програми на JavaScript можуть декомпілювати тіла функцій назад у вихідний код).

JavaScript є динамічно типізованою мовою програмування, тобто основна частина перевірок типів виконується під час виконання програми, а не під час

компіляції. У динамічній типізації значення мають типи, а змінні – ні, тому змінна може містити значення будь-якого типу.

Динамічність дозволяє простіше писати програму, але може заплутати розробника, а в деяких ситуація помилка в коді через типізацію може випадково попасти до клієнта та викликати масу помилок.

Тому над JavaScript існує безліч надбудов, які додають ті, чи інші функції до мови, а потім компілюються в JavaScript який далі виконується. Найбільш популярною надбудовою є TypeScript. [20]

Оскільки JavaScript динамічно-типізована мова програмування її не завжди зручно використовувати без доповнень та надбудов в розробці програмних застосунків. Для більш комфортної, продуктивної та якіснішої розробки застосунку було обрано TypeScript

TypeScript — це строго типізована мова програмування, яка ґрунтується на JavaScript і надає кращі інструменти будь-якого масштабу.

TypeScript додає до JavaScript додатковий синтаксис для підтримки більш тісної інтеграції з редактором. Дозволяє ловити помилки на ранніх стадіях розробки.

Код TypeScript перетворюється на JavaScript, який виконується будь-де, де працює JavaScript: у браузері, на NodeJS або Deno.

TypeScript розуміє JavaScript і використовує визначення типу, щоб надати чудові інструменти без додаткового коду. [21]

В силу повної зворотної сумісності адаптація наявних застосунків з JavaScript на TypeScript може відбуватися поетапно, шляхом поступового визначення типів. Підтримка динамічної типізації зберігається — компілятор TypeScript успішно обробить і не модифікований код на JavaScript.

Основний принцип — будь-який код на JavaScript сумісний з TypeScript, тобто в програмах на TypeScript можна використовувати стандартні JavaScript-бібліотеки і раніше створені напрацювання. Більш того, можна залишити наявні JavaScript-проекти в незмінному вигляді, а дані про типізації розмістити у вигляді

анотацій, які можна помістити в окремі файли, які не заважатимуть розробці і прямому використанню проекту (наприклад, подібний підхід зручний при розробці JavaScript-бібліотек).

JavaScript на 2022 рік є найбільш популярною мовою програмування в світі, а TypeScript займає 4 місце [22]

На рисунку 3.19 представлено графік популярності мов програмування

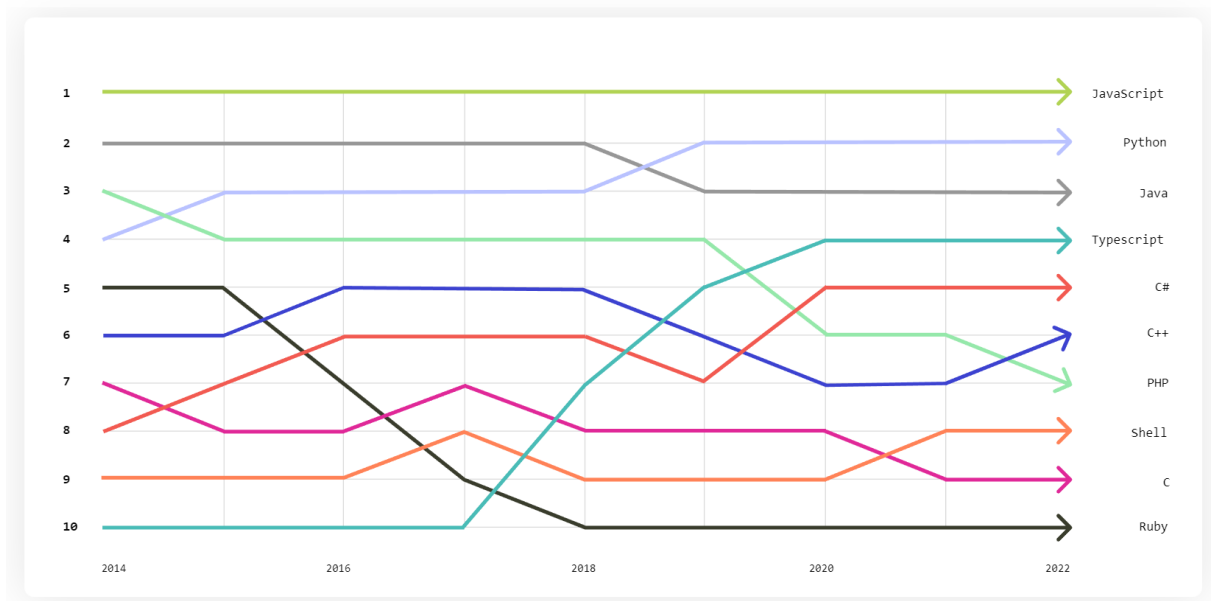


Рисунок 3.19 - Графік трендів мов програмування

Оскільки TypeScript перетворюється на JavaScript, який виконується будь-де, де працює JavaScript, а JavaScript може виконуватись як на клієнті за допомогою браузера, так і на сервері за допомогою, наприклад, NodeJS для більш ефективної розробки та перевикористання коду TypeScript було обрано як мову програмування і для серверної частини також.

Платформою для розробки серверної частини було обрано NodeJS

NodeJS — платформа, побудована на JavaScript-рушієві V8, з відкритим кодом для виконання високопродуктивних мережових застосунків, написаних мовою JavaScript.

З відкритим кодом означає, що кожна людина може запропонувати свої зміни, які надалі будуть перевірені та можуть потрапити в наступну версію.

Раніше, до створення NodeJS, JavaScript застосовувався для обробки даних в браузері користувача. NodeJS надав можливість виконувати JavaScript-скрипти на сервері та відправляти користувачеві результат їхнього виконання. Таким чином NodeJS перетворила JavaScript з вузькоспеціалізованої мови для браузерів в мову загального призначення. [23]

Для швидкого тестування застосунку в різних версіях NodeJS було обрано Node Version Manager.

Node Version Manager (далі - npm) – контролер версій NodeJS, створений для швидкої розробки та тестування застосунку без необхідності встановлювати нову чи стару версію NodeJS для кожного запуску. [24]

Для управління залежностями NodeJS було обрано npm який основний над Node Package Manager.

Node Package Manager - менеджер пакетів, що входить до складу NodeJS, найбільший у світі реєстр програмного забезпечення. Розробники пакетів з відкритим кодом з усіх континентів використовують npm для спільного використання та запозичення пакетів, а багато організацій також використовують npm для керування приватною розробкою.

Node Package Manager складається з клієнта командного рядка, який взаємодіє з віддаленим реєстром. Це дозволяє користувачам користуватися модулями JavaScript та розповсюджувати їх. В головному реєстрі npm доступно понад 477 000 пакунків. Реєстр не має процедури перевірки, а це означає, що знайдені там пакунки можуть бути низькоякісними або небезпечними. Натомість Node Package Manager спирається на звіти користувачів, щоб видаляти пакунки, якщо вони порушують політику безпеки (є незахищеними, зловмисними або низькоякісними). npm показує статистику, включаючи кількість завантажень та кількість пакунків, щоб допомогти розробникам оцінювати якість пакетів.

Node Package Manager може управляти пакунками, які є локальними залежностями певного проекту, а також глобально інсталюваними інструментами JavaScript. При використанні npm як менеджера залежності для локального проекту, можна встановити одною командою всі залежності проекту через файл package.json. У файлі package.json кожна залежність може визначати діапазон дійсних версій, використовуючи схему семантичної версії, що дозволяє розробникам автоматично оновлювати свої пакети, одночасно уникаючи небажаних змін. [25]

pnpm(далі pnpm) - це новий менеджер пакунків для JavaScript, створений на основі npm для спрощення процесу інсталяції пакетів у програмах вузла. PNPM є альтернативою NPM. Він дотримується тих же принципів, що й NPM, але має деякі додаткові функції, які роблять його потужнішим, ніж його попередник.

На рисунку 3.20 представлено візуалізацію роботи PNPM

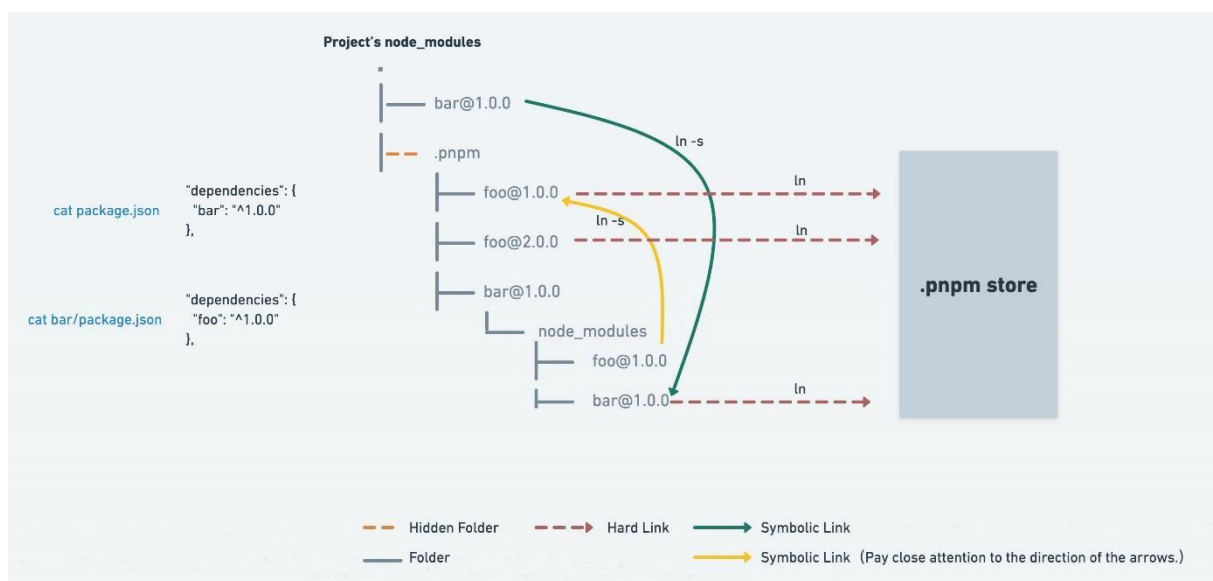


Рисунок 3.20 - Візуалізація роботи pnpm

Основною мотивацією створення pnpm була економія дискового простору та підвищення швидкості встановлення. Під час використання npm, якщо у розробника є 100 проектів, які використовують якусь залежність, буде 100 копій

цієї залежності, збережених на диску. За допомогою npm залежність зберігатиметься в сховищі з адресою вмісту, тому:

- якщо в проєкті використовуються різні версії залежності, до сховища додаються лише ті файли, які відрізняються версія від версії. Наприклад, якщо він містить 100 файлів, а нова версія містить зміни лише в одному з цих файлів, npm update до сховища буде додано лише 1 новий файл, замість того, щоб клонувати всю залежність лише для єдиної зміни;

- усі файли зберігаються в одному місці на диску. Коли пакети інстальовано, їхні файли жорстко пов'язуються з цього єдиного місця, не займаючи додаткового місця на диску. Це дає змогу поділитися залежностями однієї версії між проєктами.

Як наслідок, економія великої кількості місця на диску, пропорційна кількості проєктів і залежностей, а також набагато швидше встановлення. [26]

Для написання будь-якого веб-сайту, чи то сервісу або ж додатку в будь-якому випадку використовується мова гіпертекстової розмітки.

Мова гіпертекстової розмітки (далі HTML) – найбільш фундаментальний будівельний блок для всього вебу. HTML визначає значення та структуру веб-контенту. Інші технології окрім HTML, як правило, використовуються для опису вигляду / презентації веб-сторінки (CSS) або функціональності / поведінки (JavaScript).

"Гіпертекст" відноситься до посилань, які з'єднують веб-сторінки між собою, як в межах одного веб-сайту, так і між веб-сайтами. Посилання є фундаментальним аспектом Інтернету. Завантажуючи вміст в Інтернет та посилаючись на сторінки, створені іншими людьми, ви стаєте активним учасником всесвітньої павутини.

HTML використовує "розмітку" для анотування тексту, зображень та іншого контенту для відображення веб-браузером. [27]

Для розробки клієнтської частини сервісу було обрано frontend-фреймворк Angular

Фронтенд (далі frontend) – це публічна частина web-додатків (веб-сайтів), з якою користувач може взаємодіяти і контактувати напряму. У Frontend входить

відображення функціональних завдань призначеного для користувача інтерфейсу, що виконуються на стороні клієнта, а також обробка запитів користувачів. Frontend – це все те, що бачить користувач при відкритті web-сторінки. [28]

Фреймворк (далі framework) — це програмна оболонка, яка дозволяє спростити і прискорити вирішення типових завдань, властивих конкретній мові програмування. Фреймворк використовують, оскільки це зручно. Набагато простіше створювати проект, оскільки він уже має свою структуру.

Головна ціль фреймворку, дати розробнику комфортне середовище для проекту з великим та масштабованим функціоналом.

Framework використовує патерн проектування MVC (model-view-controller – «Модель-подання-контролер»). MVC – це обов’язкова умова для організації коду чи компонентів. Завдання MVC – вирішити проблему проектування, що виникла у робочому рішенні. На основі MVC створено різні види фреймворків, наприклад Angular або ж Vue.js. [29]

Angular - це фреймворк для дизайну та розробки додатків, який дозволяє створювати ефективні та складні односторінкові(SPA) застосунки. [30]

Односторінкові застосунки, або Single Page Application (далі SPA) - це реалізація веб-додатку таким чином, щоб він завантажував лише один веб-документ, а потім оновлював вміст тіла цього одного документа за допомогою JavaScript коли потрібно показати різний вміст.

Це дозволяє користувачам використовувати веб-сайти, не завантажуючи цілком нові сторінки з сервера, що може призвести до покращення продуктивності та більш швидшого відклику сайту, але при цьому виникає деякий недолік, такий як погіршення SEO, більше зусиль для збереження стану застосунку, реалізації навігації та проведення вимірювань продуктивності. [31]

На рисунку 3.21 зображена візуалізація традиційного підходу та SPA щодо життєвого циклу веб-сторінки [32]

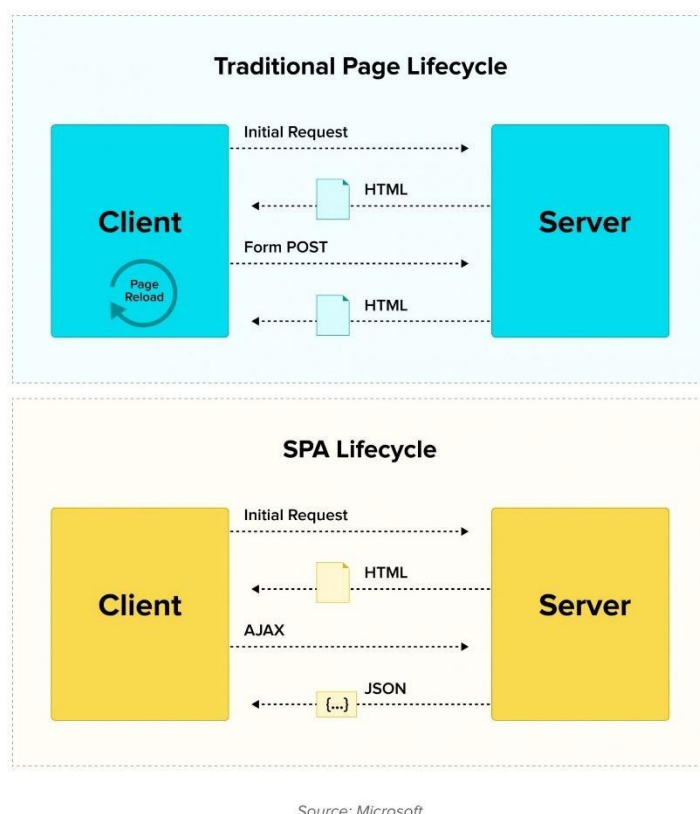


Рисунок 3.21 – Візуалізація роботи традиційного підходу та SPA щодо життєвого циклу веб-сторінки

Angular має, в порівнянні з React та Vue, деякі відмінності.

React – бібліотека JavaScript для створення користувацьких інтерфейсів. React використовує декларативний, базований на компонентах, підход для створення інтерфейсів. [33]

Vue - це фреймворк, який працює на JavaScript, створений для розробки користувацьких інтерфейсів. Він працює на базі звичайного HTML, CSS та JavaScript, з можливостями декларативно програмувати користувацькі інтерфейси будь-якої складності на основі компонентів. [34]

На рисунку 3.22 зображений приклад коду однофайлового компоненту на Vue

```
<script setup>
import { ref } from 'vue'
const greeting = ref('Привіт, світе!')
</script>

<template>
  <p class="greeting">{{ greeting }}</p>
</template>

<style>
.greeting {
  color: red;
  font-weight: bold;
}
</style>
```

Рисунок 3.22 – Приклад однофайлового компоненту на фреймворку Vue

JSX — це синтаксичне розширення для JavaScript, яке дозволяє писати HTML-подібну розмітку всередині файлу JavaScript. Хоча існують інші способи написання компонентів, більшість розробників React віддають перевагу лаконічності JSX, і більшість кодових баз використовують його. [35]

На рисунку 3.23 зображений код для створення React компоненту використовуючи JSX

```

const person = {
  name: 'Gregorio Y. Zara',
  theme: {
    backgroundColor: 'black',
    color: 'pink'
  }
};

export default function TodoList() {
  return (
    <div style={person.theme}>
      <h1>{person.name}'s Todos</h1>
      
      <ul>
        <li>Improve the videophone</li>
        <li>Prepare aeronautics lectures</li>
        <li>Work on the alcohol-fuelled engine</li>
      </ul>
    </div>
  );
}

```

Рисунок 3.23 – Приклад коду створення компоненту для бібліотеки React використовуючи JSX

Angular є html-first фреймворком, тобто розробник спочатку пише html, а потім вставив в нього директиви Angular та описує за допомогою TypeScript як воно повинно працювати. Vue також є html-first фреймворком.

Приклад html-коду для Angular зображений на рисунку 3.24.

```

<ng-container *ngIf="show">
  <div *ngFor="let thing of stuff">
    {{thing.name}}
    <span>{{thing.comment}}</span>
  </div>
</ng-container>

```

Рисунок 3.24 – Приклад html-коду для фреймворку Angular

React же є javascript-first бібліотекою. Доказом цього є приклад коду, який зображений на рисунку 3.25 написаного для React без використання JSX. Як видно з зображення це чистий javascript код який викликає функції бібліотеки React напряду без написання html, а лише вказуючи назви тегів які потрібно створити.

```
import { createElement } from 'react';

function Greeting({ name }) {
  return createElement(
    'h1',
    { className: 'greeting' },
    'Hello ',
    createElement('i', null, name),
    '. Welcome!'
  );
}

export default function App() {
  return createElement(
    Greeting,
    { name: 'Taylor' }
  );
}
```

Рисунок 3.25 – Приклад коду для бібліотеки React написаного без використання JSX

Прив'язка даних (Data binding) - це процес зв'язку двох джерел даних та їх синхронізації. З допомогою прив'язки даних, зміна елемента в наборі даних автоматично оновлюється в пов'язаному наборі даних.

Прив'язка з одного боку або одностороння прив'язка (One-way binding) - це відносно простий тип прив'язки даних. Зміни, які відбуваються в постачальнику даних, автоматично оновлюються у споживачах даних, але не навпаки.

Прив'язка з двох боків або двостороння прив'язка (Two-way binding) - це випадок, коли зміни, які відбуваються або в постачальнику даних, або в споживачах даних, автоматично оновлюють інші джерела даних. [36]

Angular підтримує обидва види прив'язки даних, та має три типи:

- одностороння прив'язка даних від джерела даних до цільового перегляду, використовується для передачі параметрів(Inputs) компонентам та для відображення змінних на сторінці;

- одностороння прив'язка даних від відображаємої цілі до джерела даних. Використовується для деяких подій javascript та подій компонентів(Outputs);

- двостороння прив'язка даних. [37]

Vue, як і Angular, підтримує обидва види прив'язки даних

React же підтримує тільки одно-сторонню прив'язку даних, від джерела даних до цільового перегляду або ж компонентам(props). Тому для отримання даних від користувача в React використовують стандартні javascript правила.

Великою перевагою Angular над React та Vue є Angular CLI.

Angular CLI – текстовий інтерфейс користувача, який надає можливість з легкістю створювати додаток, запускати, тестувати та створювати нові компоненти, сервіси, директиви та інше. Це відповідає кращим сучасним тенденціям та є дуже зручним при створенні хоч скільки то великого проєкту.

На рисунку 3.26 представлена консольна команда Angular CLI для генерації модуля Graph

```
ng generate module Graph
```

Рисунок 3.26 – Консольна команда Angular CLI для генерації модуля Graph

Декоратор — це структурний патерн проєктування, що дає змогу динамічно додавати об'єктам нову функціональність, загортаючи їх у корисні «обгортки». [38]

React компонентом є одна функція, або ж клас.

Vue компонентом є єдиний файл.

Angular компонентом є клас з декоратором '@Component(...)' в якому розробник може вказати наступні параметри(основні які найбільш часто використовуються):

- selector - назва за якою інші компоненти зможуть використовувати цей;
- template або ж templateUrl – HTML-шаблон або ж посилання на файл в якому цей шаблон знаходиться;
- styleUrls або ж styles – посилання на файли стилів або ж масив стилів в вигляді строки.

На рисунку 3.27 представлений приклад стандартного Angular компоненту

```
import { Component } from '@angular/core'

@Component({
  selector: 'counter-categories-table',
  templateUrl: './categories-table.component.html',
  styleUrls: ['./categories-table.component.scss']
})
export class CategoriesTableComponent {

}
```

Рисунок 3.27 – Приклад стандартного Angular компоненту

Однією з головних відмінностей Angular від React та Vue є наявність різних типів декораторів.

Якщо в React є функції-компоненти React, які містять в собі JSX та використовують React, а також всі інші звичайні JavaScript функції. Або ж в Vue є одно файлові компоненти та всі інші звичайні JavaScript функції.

Впровадження залежностей (DI) — це частина фреймворку Angular, яка надає компонентам доступ до сервісів та інших ресурсів. Angular надає розробникам можливість вставити службу в компонент, щоб надати цьому компоненту доступ до служби.

Декоратор '@Injectable()' визначає клас як службу в Angular і дозволяє Angular вставляти його в компонент як залежність. Подібним чином декоратор '@Injectable()' вказує, що компонент, клас, канал або NgModule мають залежність від служби.

Для будь-якої залежності, яка вам потрібна у вашій програмі, ви повинні зареєструвати постачальника в інжекторі програми, щоб інжектор міг використовувати постачальника для створення нових екземплярів. Для послуги постачальник зазвичай є самим класом послуги. [39]

В Angular є досить велика кількість різних типів класів, а оскільки стандартом в програмуванні є один клас на файл дуже зручно створювати красиву, зручну, а саме головне ефективну для розробки структуру застосування. Наприклад, в Angular є наступні типи класів:

- module – '@NgModule' – Модулі в Angular – це контейнери для об'єднання коду в блоки, призначеного для домену програми, робочого процесу або тісно пов'язаного набору можливостей. Вони можуть містити компоненти, постачальників послуг та інші кодові файли, область дії яких визначається NgModule, що містить. Вони можуть імпортувати функції, які експортуються з інших NgModules, і експортувати вибрані функції для використання іншими NgModules; [40]

- component – '@Component' - Компоненти є основним будівельним блоком для додатків Angular; [41]

- service – '@Injectable' – Дозволяє створити сервіс, який потім можна використовувати в іншому сервісі, компоненті, експортувати в модулі для використання в інших модулях та інше;

- pipe – '@Pipe' - Клас, якому передую декоратор '@Pipe{}' і який визначає функцію, яка перетворює вхідні значення у вихідні значення для відображення у поданні. Angular визначає різні pipe, і розробник може створити нові; [42]

- directive – '@Directive' – Директиви атрибути в Angular дозволяють змінювати зовнішній вигляд або поведінку елементів DOM і компонентів Angular. [43]

В більшості своєму Angular використовується для створення великих багато навантажених проєктів, наприклад Google, які і є розробниками Angular, саме тому цей фреймворк має всі необхідні інструменти для швидкої та зручної розробки.

Зважаючи на всі вище перелічені відмінності Angular від інших фреймворків для виконання сервісу було обрано саме його

Для пришвидшення розробки стилізації сервісу було обрано Tailwindcss

Tailwindcss – перший утилітарний фреймворк CSS наповнений класами, за допомогою яких можна створити будь-який дизайн безпосередньо в мові гіпертекстової розмітки.

Tailwindcss дозволяє швидко створювати сучасні веб-сайти, не відриваючись від HTML [44]

На рисунку 3.28 відображений приклад використання tailwindcss для оформлення кнопки «Купити зараз»

```
<button class="h-10 px-6 font-semibold rounded-md  
bg-black text-white" type="submit">  
    Купити зараз  
</button>
```

Рисунок 3.28 – Приклад стандартного Angular компоненту

Для більш зручної роботи з реактивними даними, тобто даними, що можуть асинхронно змінюватись з плином часу, а також для роботи бібліотеки яка займається загальним станом всього сервісу, було взято бібліотеку RxJs.

RxJS — це бібліотека для реактивного програмування з використанням Observables, щоб полегшити створення асинхронного коду або коду на основі зворотного виклику.

RxJS є переписаним Reactive-Extensions/RxJS із кращою продуктивністю, кращою модульністю, кращими стеками викликів із можливістю налагодження, зберігаючи в основному зворотну сумісність із деякими критичними змінами, які зменшують поверхню API.

RxJS — це бібліотека для створення асинхронних і програм які базуються на подіях за допомогою спостережуваних послідовностей.

RxJS надає один основний тип, Observable, супутникові типи (Observer, Schedulers, Subjects) і оператори, натхненні методами Array (map, filter, reduce, every тощо), щоб дозволити обробку асинхронних подій як колекцій.

ReactiveX поєднує шаблон проектування Observer із шаблоном проектування Iterator і функціональне програмування з колекціями, щоб задовольнити потребу в ідеальному способі керування послідовністю подій. Основні концепції в RxJS, які вирішують асинхронне керування подіями:

Observable(Спостережувані) - представляє ідею викликаної колекції майбутніх значень або подій.

Observer(Спостерігач) - це набір зворотних викликів, який знає, як прослуховувати значення, надані Observable.

Subscription(Підписка) - представляє виконання Observable, в першу чергу корисна для скасування виконання.

Operators(Оператори) - це чисті функції, які забезпечують функціональний стиль програмування роботи з колекціями за допомогою таких операцій, як map, filter, concat, reduce тощо.

Subject(Предмет) - еквівалент Angular EventEmitter і єдиний спосіб групової передачі значення або події кільком спостерігачам.

Schedulers(Планувальники) - це централізовані диспетчери для керування паралельністю, що дозволяє нам координувати, коли обчислення відбуваються, наприклад, `setTimeout` або `requestAnimationFrame` або інші.[45]

На рисунку 3.29 зображений приклад роботи RxJs.

```
const observable = Rx.Observable.create(function (observer) {
  observer.next(1);
  observer.next(2);
  observer.next(3);

  setTimeout(() => {
    observer.next(4);
    observer.complete();
  }, 1000);
});

console.log('just before subscribe');

observable.subscribe({
  next: x => console.log('got value ' + x),
  error: err => console.error('something wrong occurred: ' + err),
  complete: () => console.log('done'),
});

console.log('just after subscribe');

/*
just before subscribe
got value 1
got value 2
got value 3
just after subscribe
got value 4
done
*/
```

Рисунок 3.29 – Приклад роботи RxJs

Для зберігання глобального стану сервісу було обрано бібліотеки `@ngrx/store` разом з `@ngrx/effects` та допоміжну `@ngrx/store-devtools` для тестування та відладки.

`@ngrx/store` забезпечує керування станом для створення супроводжуваних явних додатків за допомогою використання єдиного стану та дій для вираження змін стану.

`@ngrx/store` — це бібліотека для керування глобальним станом застосунку на базі RxJS для програм Angular, натхненне Redux.

@ngrx/store — це контейнер із контрольованим станом, розроблений, щоб допомогти писати продуктивні, узгоджені програми поверх Angular.

Безпека типів підтримується в усій архітектурі з опорою на компілятор TypeScript для коректності програми. На додаток до цього, суворість безпеки типів у NgRx і використання шаблонів добре підходить для створення коду вищої якості.

@ngrx/store побудовано на єдиній незмінній структурі даних, що робить виявлення змін відносно простим завданням за допомогою Angular OnPush стратегії. @ngrx/store також надає API для створення мемоізованих функцій вибору, які оптимізують отримання даних із стану.

Використовуючи @ngrx/effects and @ngrx/store, будь-які побічні ефекти взаємодії із зовнішніми ресурсами, такими як мережеві запити або веб-сокети, а також будь-яка бізнес-логіка, можуть бути ізольовані від інтерфейсу користувача. Ця ізоляція дозволяє використовувати більш чисті та прості компоненти та підтримує принцип єдиної відповідальності.

Нормалізуючи зміни стану та передаючи їх через спостережувані, NgRx забезпечує можливість серіалізації та забезпечує передбачуване збереження стану. Це дозволяє зберегти стан у зовнішній пам'яті, наприклад localStorage.

Це також дозволяє перевіряти, завантажувати, завантажувати та відправляти дії з @ngrx/store-devtools.

Оскільки @ngrx/store використовує чисті функції для зміни та вибору даних із стану, а також здатність ізолювати побічні ефекти від інтерфейсу користувача, тестування стає дуже простим. NgRx також надає тестові ресурси, такі як provideMockStore і provideMockActions для ізольованих тестів, і загалом кращий досвід тестування. [46]

@ngrx/effects — це модель побічних ефектів на основі RxJS для @ngrx/store.

@ngrx/effects використовують потоки, щоб забезпечити нові джерела дій для зменшення стану на основі зовнішніх взаємодій, таких як мережеві запити, повідомлення веб-сокетів і часові події.

У сервісному додатку Angular компоненти відповідають за взаємодію із зовнішніми ресурсами безпосередньо через сервіси. Натомість ефекти забезпечують спосіб взаємодії з цими службами та ізолюють їх від компонентів. Ефекти – це те, де ви виконуєте такі завдання, як отримання даних, довгострокові завдання, які створюють кілька подій, та інші зовнішні взаємодії, де вашим компонентам не потрібні чіткі знання про ці взаємодії.

Ефекти ізолюють побічні ефекти від компонентів, дозволяючи використовувати більш чисті компоненти, які вибирають стан і дії диспетчеризації.

Ефекти — це довготривалі служби, які прослуховують спостережувану кожну дію, що надсилається з store.

Ефекти фільтрують ці дії на основі типу дії, яка їх цікавить. Це робиться за допомогою оператора.

Ефекти виконують завдання, які є синхронними або асинхронними, і повертають нову дію. [47]

На рисунку 3.30 зображений приклад коду `@ngrx/effect`

```

loadStatistic$ = createEffect(() =>
  this.actions$.pipe(
    ofType(StatisticActions.load),
    withLatestFrom(this.store.pipe(select('statistic'))),
    mergeMap(([params, statisticValue]) => {
      if (statisticValue.length === 0 || params.force) {
        this.store.dispatch(
          StatisticStatusActions.set({
            status: StatisticStatusTypes.StatusState.SYNCHRONIZATION
          })
        )
      }

      return this.api.getAllStatisticRecords().pipe(
        mergeMap(value => {
          return [
            StatisticStatusActions.set({
              status: StatisticStatusTypes.StatusState.SYNCHRONIZED
            }),
            StatisticSyncActions.set({ statistic: value })
          ]
        })
      ),
      catchError(() => {
        this.store.dispatch(
          StatisticStatusActions.set({
            status: StatisticStatusTypes.StatusState.ERROR
          })
        )
      })
    })
  )
  return EMPTY
})
return EMPTY
})
)

```

Рисунок 3.30 – Приклад коду @ngrx/effect

Для роботи з svg іконками було обрано angular-svg-icon

angular-svg-icon — це сервіс і компонент Angular 15, який надає засоби для вбудовування файлів SVG, щоб можна було легко стилізувати їх за допомогою CSS і коду.

Сервіс надає реєстр значків, який завантажує та кешує SVG, індексований за його url. Компонент відповідає за відображення SVG. Після отримання svg із реєстру він клонує SVGElement і SVG у внутрішній HTML компонента. [48]

На рисунку 3.31 зображений приклад використання angular-svg-icon

```

<svg-icon
  class="inline-block w-5 h-5"
  [ngClass]="{
    'stroke-gray-400': statusIsNotSynchronized,
    'stroke-amber-400': statusIsSynchronization,
    'stroke-red-400': statusIsError
  }"
  src="assets/icons/sync.svg"
></svg-icon>

```

Рисунок 3.31 – Приклад використання angular-svg-icon

Для відображення анімованих svg було взято бібліотеку lottie-web та зв'язку її з Angular – ngx-lottie

Lottie — це бібліотека для Android, iOS, Web і Windows, яка аналізує анімацію Adobe After Effects, експортовану як JSON за допомогою Bodymovin, і рендерить її на мобільних пристроях і в Інтернеті! [49]

Lottie-web – Адаптована версія lottie для написання веб

Ngx-lottie - Мінімально настраюваний компонент Angular зі стабільною продуктивністю для відтворення анімації After Effects. [50]

На рисунку 3.32 зображений приклад використання lottie в Angular

```

<div class="lottie-animation w-8">
  <ng-lottie
    [options]="getAnimateOptions(item.id)"
    (animationCreated)="onAnimate($event)"
  ></ng-lottie>
</div>

```

Рисунок 3.32 – Приклад використання lottie

Для створення функціоналу зміни порядку категорій на сторінці клієнтської частини категорій було використано @angular/cdk/drag-drop

Набір для розробки компонентів, або ж Component Dev Kit (CDK) — це набір примітивів поведінки для створення компонентів інтерфейсу користувача. [51]

Модуль `@angular/cdk/drag-drop` надає розробникам спосіб легкого та декларативного створення інтерфейсів перетягування та скидання з підтримкою вільного перетягування, сортування в списку, перенесення елементів між списками, анімації, сенсорних пристроїв, спеціального перетягування ручки, попередні перегляди та заповнювачі, на додаток до горизонтальних списків і блокування вздовж осі. [52]

Для створення графіку на сторінці статистики було обрано бібліотеку Chart.js

Chart.js - проста, але гнучка бібліотека діаграм JavaScript для сучасного Інтернету

Серед багатьох бібліотек діаграм для розробників додатків JavaScript Chart.js наразі є найпопулярнішою згідно зі зірками GitHub (~60 000) і завантаженнями npm (~2 400 000 щотижня).

Chart.js був створений і оголошений у 2013 році, але з того часу пройшов довгий шлях. Він має відкритий вихідний код, ліцензований за дуже дозволеною ліцензією MIT і підтримується активною спільнотою.

Chart.js надає набір часто використовуваних типів діаграм, плагінів і параметрів налаштування. Окрім розумного набору вбудованих типів діаграм, ви можете використовувати додаткові типи діаграм, які підтримуються спільнотою. Крім того, можна об'єднати кілька типів діаграм у змішану діаграму (по суті, змішуючи кілька типів діаграм в одну на одному полотні). Chart.js легко налаштовується за допомогою користувальницьких плагінів для створення анотацій, масштабування або функцій перетягування. [53]

Для створення авторизації за допомогою Google аккаунту було використано Google identity та `@abacritt/angularx-social-login`

Google Identity Platform(далі Identity Platform) – це платформа керування ідентифікацією та доступом клієнтів (CIAM), яка допомагає організаціям додавати

функції керування ідентифікацією та доступом до своїх програм, захищати облікові записи користувачів і впевнено масштабуватись у Google Cloud.

За допомогою платформи Identity Platform розробники можуть легко додати широко поширену, зручну та настроювану службу автентифікації до своїх веб- і мобільних програм, щоб вони могли зосередитися на створенні своєї програми чи послуги.

Платформа Identity Platform може допомогти захистити користувачів додатків та запобігти захопленню облікових записів, пропонуючи багатофакторну автентифікацію (MFA) та інтеграцію з розвідкою Google для захисту облікових записів. [54]

@abacritt/angularx-social-login – Модуль для соціального входу та автентифікації для Angular. Підтримує автентифікацію в Google, Facebook, Amazon та Microsoft.

Клієнтська частина сервісу має 3 основні сторінки та 1 додаткову

В шапці клієнтської частини сервісу знаходиться логотип при натисненні на який користувач перейде на головну сторінку, меню, а також інформація про користувача, а саме його Ім'я та Прізвище, а також аватар.

На рисунку 3.33 продемонстрована шапка сервісу



Рисунок 3.33 – Демонстрація шапки сервісу

Якщо натиснути на інформацію про користувача з'явиться меню, в якому на даний момент можна вийти з аккаунту

На рисунку 3.34 продемонстровано меню користувача

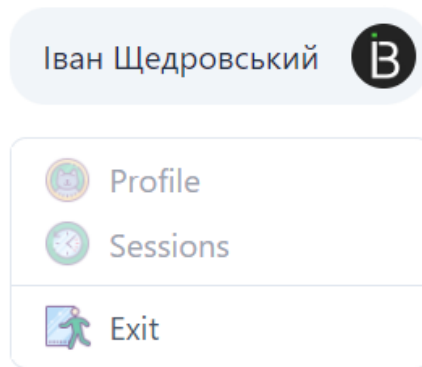


Рисунок 3.34 – Демонстрація меню користувача

Футор, або ж нижня панель сервісу має інформацію про поточну версію застосунку, посилання на [github](#) розробника, а також посилання на [tailwindcss](#) та [icons8](#), сервіси які були використані при розробці, та які по ліцензійній угоді потрібно указувати. На рисунку 3.35 продемонстрований футер сервісу

v1.1.0 next

Created by [Itlaitoff](#)



Рисунок 3.35 – Демонстрація футера сервісу

Домашня сторінка представляє собою форму для вводу нових даних статистики обираючи раніше створені категорії, вводячи коментар. Також для форми є додаткові опції, які можна побачити натиснувши на кнопку “Additional”. На рисунку 3.36 продемонстрована головна сторінка застосунку

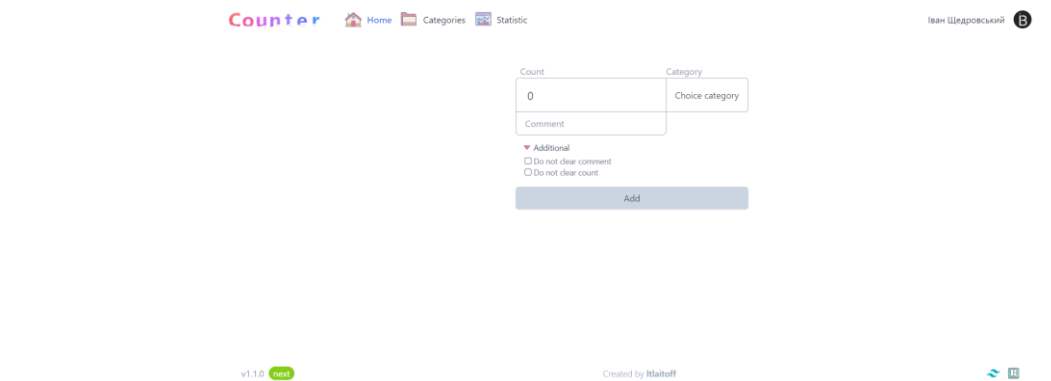


Рисунок 3.36 – Демонстрація головної сторінки застосунку

Сторінка категорій представляє собою дві кнопки та таблицю з категоріями користувача.

Перша кнопка «Add new category» при натисненні відкриває форму для додавання нової категорії.

При натисненні на другу кнопку відбудеться примусове перезавантаження даних з API без перезавантаження сторінки. А також ця кнопка показує поточний статус категорій, всього їх 4: Не синхронізовано, синхронізація, помилка та синхронізовано.

Таблиця має в собі 4 стовпця:

- перший стовпець відповідає за колір категорії;
- другий за її назву;
- третій за коментар;
- четвертий за тип величин для зручності користувачу.

На рисунку 3.37 зображена вся сторінка категорій

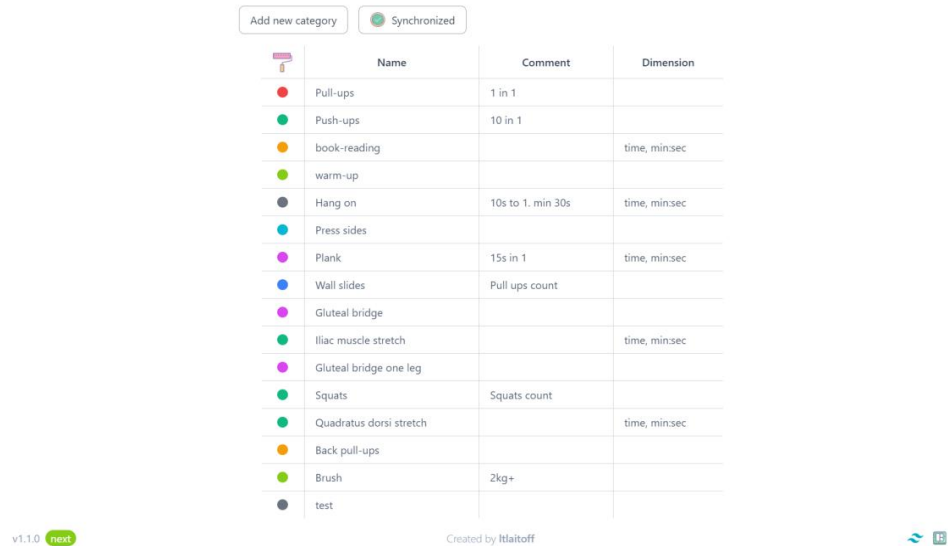


Рисунок 3.37 – Демонстрація всієї сторінки категорій

На сторінці статистики зображений графік та таблиця даних. На рисунку 3.38 представлений приклад графіку статистики на сторінці статистики

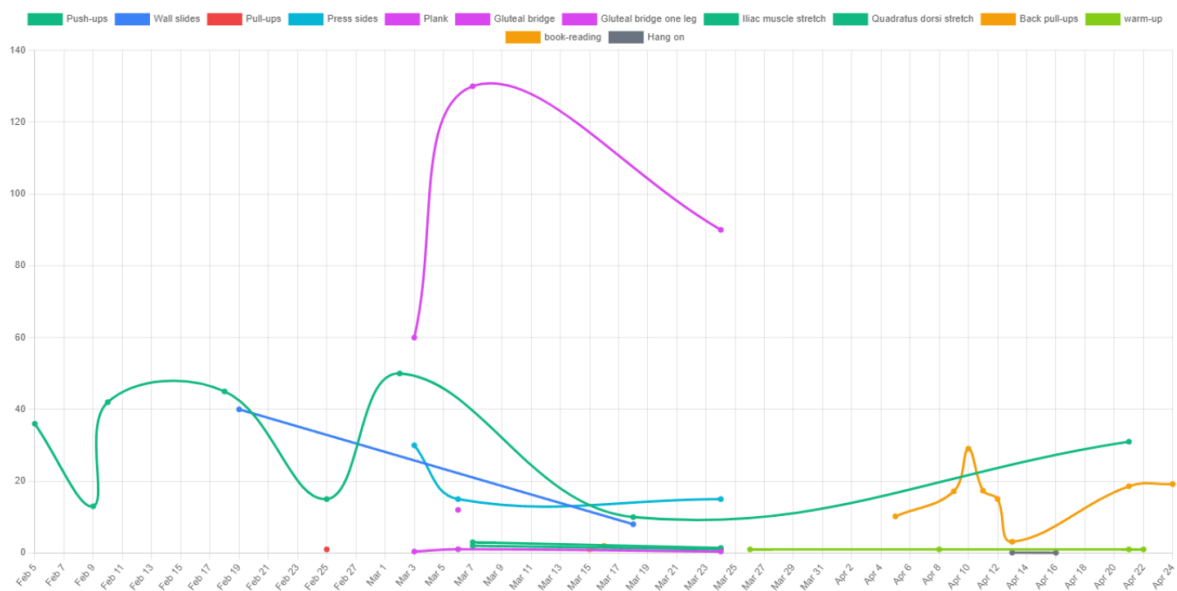


Рисунок 3.38 – Приклад графіку статистики

При наведені на рядок таблиці статистики з'являються дві кнопки. За допомогою кнопки яка позначена як корзина можна видалити запис. За допомогою кнопки, яка позначена як ручка – можна змінити запис в відповідний формі. На рисунку 3.39 представлений приклад таблиці статистики з наведенням миші на рядок

Synchronized

Count	Comment	Category	Date
10		● Push-ups	Feb 5, 1:24:03 AM 2023
10		● Push-ups	Feb 5, 11:41:07 AM 2023
16		● Push-ups	Feb 5, 4:50:10 PM 2023
13		● Push-ups	Feb 9, 1:23:57 AM 2023
12		● Push-ups	Feb 10, 7:02:17 PM 2023
15		● Push-ups	Feb 10, 7:15:49 PM 2023
15		● Push-ups	Feb 10, 11:11:42 PM 2023
15		● Push-ups	Feb 18, 11:17:57 PM 2023
15		● Push-ups	Feb 18, 11:39:05 PM 2023
15		● Push-ups	Feb 18, 11:57:49 PM 2023
10		● Wall slides	Feb 19, 11:36:47 PM 2023
15		● Wall slides	Feb 19, 11:54:12 PM 2023
15		● Wall slides	Feb 19, 11:59:07 PM 2023
1		● Pull-ups	Feb 25, 1:55:43 PM 2023
15		● Push-ups	Feb 25, 1:58:16 PM 2023
10	1/10	● Push-ups	Mar 2, 10:44:53 PM 2023
10	2/10	● Push-ups	Mar 2, 10:50:02 PM 2023

Рисунок 3.39 – Приклад таблиці статистики з наведенням миші на рядок таблиці

На сторінці авторизація представлена одна кнопка для входу в аккаунт за допомогою Google Identity

На рисунку 3.40 представлений вигляд сторінки авторизації

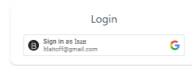


Рисунок 3.40 – Вигляд сторінки авторизації

Для створення серверної частини сервісу було обрано фреймворк NestJS

Nest (NestJS) — це фреймворк для створення ефективних, масштабованих серверних програм Node.js. Він використовує прогресивний JavaScript, створений і повністю підтримує TypeScript (все ще дозволяє розробникам кодувати на чистому JavaScript) і поєднує елементи ООП (об’єктно-орієнтоване програмування), FP (функціональне програмування) і FRP (функціональне реактивне програмування).

Під капотом Nest використовує надійні фреймворки HTTP-сервера, такі як Express (за замовчуванням).

Nest забезпечує рівень абстракції, вищий за ці звичайні фреймворки Node.js (Express/Fastify), але також надає їхні API безпосередньо розробнику. Це дає розробникам свободу використовувати незліченну кількість сторонніх модулів, які доступні для основної платформи.

В останні роки, завдяки Node.js, JavaScript став «лінгва франка» Інтернету для зовнішніх і внутрішніх програм. Це призвело до появи чудових проєктів, таких як Angular, React і Vue, які покращують продуктивність розробників і дозволяють створювати швидкі, тестовані та розширювані зовнішні програми. Однак, незважаючи на те, що існує безліч чудових бібліотек, допоміжних засобів та

інструментів для Node (і серверного JavaScript), жоден із них не вирішує ефективно головну проблему – архітектури .

Nest надає готову архітектуру додатків, яка дозволяє розробникам і командам створювати високоякісні, масштабовані, слабо пов'язані та легко підтримувані додатки. Архітектура значною мірою натхненна Angular. [55]

Під капотом Nest використовує надійні фреймворки HTTP-сервера, такі як Express (за замовчуванням).

Express - backend фреймворк для створення web-застосунків, а також для створення RESTful APIs з Node.js, є безкоштовним та знаходиться в open-source. Його називають де-факто стандартною серверною структурою для Node.js

Express - мінімалістичний та гнучкий веб-фреймворк для програм Node.js, надає широкий набір функцій для мобільних та веб-додатків. Маючи у своєму розпорядженні безліч службових методів HTTP та проміжних обробників, створити надійний API можна швидко та легко. Express надає тонкий шар фундаментальних функцій веб-застосунків, які не заважають вам працювати з давно знайомими та улюбленими вами функціями Node.js. [56]

Для розробки серверної частини сервісу було обрано саме NestJS, а не Express бо Nest надає більше можливостей для розробки та його архітектура в більшості натхнена Angular, в порівнянні з Express.

Функції проміжного програмного забезпечення(далі middleware) — це функції, які мають доступ до об'єкта запиту (req), об'єкта відповіді (res) і наступної функції проміжного програмного забезпечення в циклі запит-відповідь програми. Наступна функція проміжного програмного забезпечення зазвичай позначається змінною з іменем next.[57]

Життєвий цикл Express представляє собою набір middleware's. Життєвий цикл Express представлений на рисунку 3.41. [58]

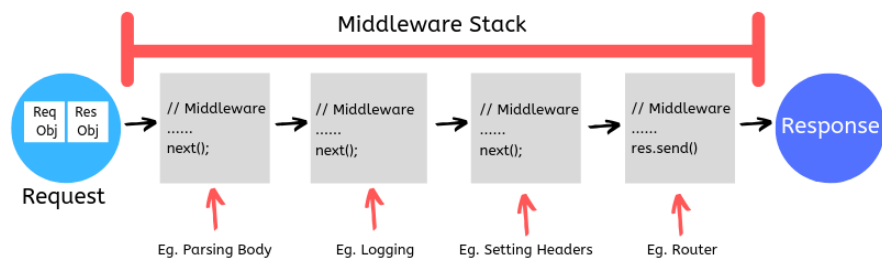


Рисунок 3.41 – Демонстрація життєвого циклу Express

Оскільки в Express фактично всі функції через які проходить http-запит є middleware це дуже не зручно та не зрозуміло для розробників, бо потрібно заходити в кожну middleware та дивитись для чого вона потрібна та що робить.

Наприклад, в розробці існують паттерни проєктування для того, щоб розробники говорили на одній мові та не потрібно було пояснювати що робить код.

Життєвий цикл Nest, зображений на рисунку 3.42, є більш складнішим та має більшу кількість елементів.[59]

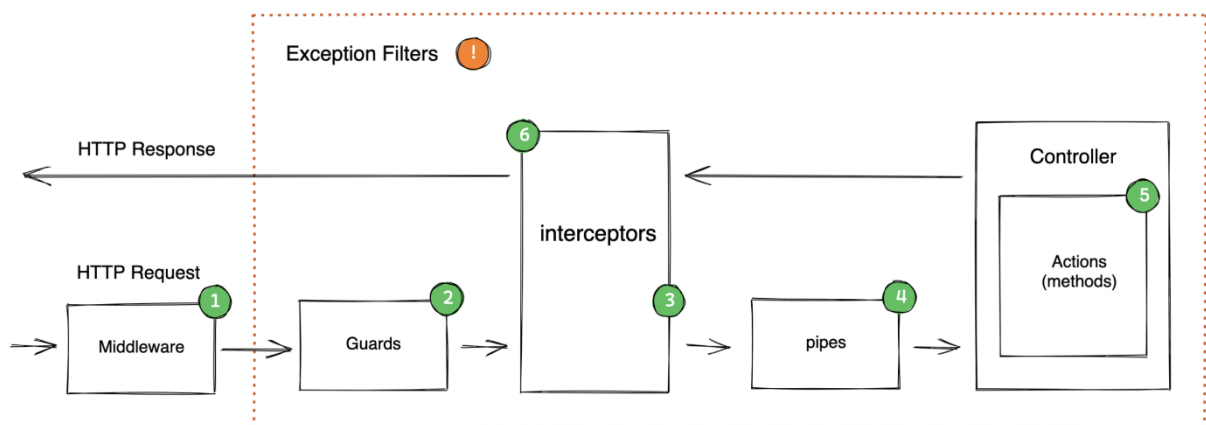


Рисунок 3.42 – Життєвий цикл Nest

Загалом життєвий цикл запиту виглядає так:

- вхідний запит;
- middleware глобального зв'язку;

- middleware пов'язане з модулем;
- глобальні guards;
- контролери guards;
- guards маршруту;
- глобальні interceptors (перед-контролер) ;
- перехоплювачі interceptors (перед-контролер) ;
- перехоплювачі маршрутів (перед-контролер) ;
- глобальні pipes;
- pipes контролера;
- pipes маршруту;
- pipes параметрів маршруту;
- контролер (обробник методу) ;
- service (якщо є) ;
- interceptor маршруту (після запиту) ;
- interceptor контролера (після запиту) ;
- глобальний interceptor (після запиту) ;
- filters винятків (маршрут, потім контролер, потім глобальний) ;
- відповідь сервера.

Middleware - проміжне програмне забезпечення виконується в певній послідовності. Спочатку Nest запускає глобально прив'язане проміжне програмне забезпечення (наприклад, проміжне програмне забезпечення, пов'язане з app.use), а потім запускає проміжне програмне забезпечення, пов'язане з модулем, яке визначається шляхом. Проміжне програмне забезпечення запускається послідовно в порядку їх зв'язування, подібно до того, як працює проміжне програмне забезпечення в Express. У випадку проміжного програмного забезпечення, зв'язаного між різними модулями, спочатку буде працювати проміжне програмне забезпечення, прив'язане до кореневого модуля, а потім проміжне програмне забезпечення — у порядку, у якому модулі додаються до масиву імпорту.

Guards - виконання охорони починається з глобальних охоронців, потім переходить до охоронців контролерів і, нарешті, до охоронців маршрутів. Як і у випадку з проміжним програмним забезпеченням, охоронці працюють у тому порядку, у якому вони зв'язані.

Interceptors - Перехоплювачі, здебільшого, дотримуються тієї самої моделі, що й охоронці, з одним недоліком: коли перехоплювачі повертають RxJS Observables, спостережувані будуть вирішені за принципом «перший, останній, вихід». Отже, вхідні запити проходять через стандартну глобальну розв'язку на рівні контролера, маршруту, але сторона відповіді запиту (тобто після повернення від обробника методу контролера) буде вирішена від маршруту до контролера до глобальної. Крім того, будь-які помилки, викинуті каналами, контролерами або службами, можна прочитати в catchError оператори перехоплювача.

Pipes - Конвеєри дотримуються стандартної глобальної послідовності зв'язування з контролером і маршрутом, з однаковими параметрами @UsePipes(). Однак на рівні параметрів маршруту, якщо у вас запущено кілька каналів, вони працюватимуть у порядку останнього параметра з каналом до першого. Це також стосується труб рівня маршруту та рівня контролера.

Filters, або ж фільтри — це єдиний компонент, який спочатку не вирішує глобальні. Натомість фільтри розв'язуються з найнижчого можливого рівня, тобто виконання починається з будь-яких фільтрів, пов'язаних з маршрутом, і продовжується на рівні контролера, а нарешті – до глобальних фільтрів. Зауважте, що винятки не можна передавати від фільтра до фільтра; якщо фільтр рівня маршруту перехоплює виняток, контролер або фільтр глобального рівня не можуть перехопити той самий виняток. Єдиний спосіб досягти такого ефекту — використовувати успадкування між фільтрами. [60]

В якості бази даних для сервісу було обрано MongoDB.

MongoDB — документо-орієнтована система керування базами даних з відкритим вихідним кодом, яка не потребує опису схеми таблиць. MongoDB займає нішу між швидкими і масштабованими системами, що оперують даними у форматі ключ/значення, і реляційними системами контролю бази даних, функціональними і зручними у формуванні запитів. MongoDB написана на мові C++ і поширюється в рамках ліцензії AGPLv3.

MongoDB підтримує зберігання документів в JSON-подібному форматі, має досить гнучку мову для формування запитів, може створювати індекси для різних збережених атрибутів, ефективно забезпечує зберігання великих бінарних об'єктів, підтримує журналювання операцій зі зміни і додавання даних в БД, може працювати відповідно до парадигми Map/Reduce, підтримує реплікацію і побудову відмовостійких конфігурацій. У MongoDB є вбудовані засоби із забезпечення шардінгу (розподіл набору даних по серверах на основі певного ключа), комбінуючи який з реплікацією даних можна побудувати горизонтально масштабований кластер зберігання, в якому відсутня єдина точка відмови (збій будь-якого вузла не позначається на роботі БД), підтримується автоматичне відновлення після збою і перенесення навантаження з вузла, який вийшов з ладу. Розширення кластера або перетворення одного сервера на кластер проводиться без зупинки роботи БД простим додаванням нових машин.

СКБД управляє наборами JSON-подібних документів, що зберігаються в бінарному форматі BSON. Зберігання і пошук файлів в MongoDB відбувається завдяки викликам протоколу GridFS. [61]

Mongoose — це бібліотека моделювання об'єктних даних (ODM) для MongoDB і Node.js. Він керує зв'язками між даними, забезпечує перевірку схеми та використовується для перекладу між об'єктами в коді та представленням цих об'єктів у MongoDB.

На рисунку 3.43 представлено візуалізацію роботи NodeJS та MongoDB через Mongoose

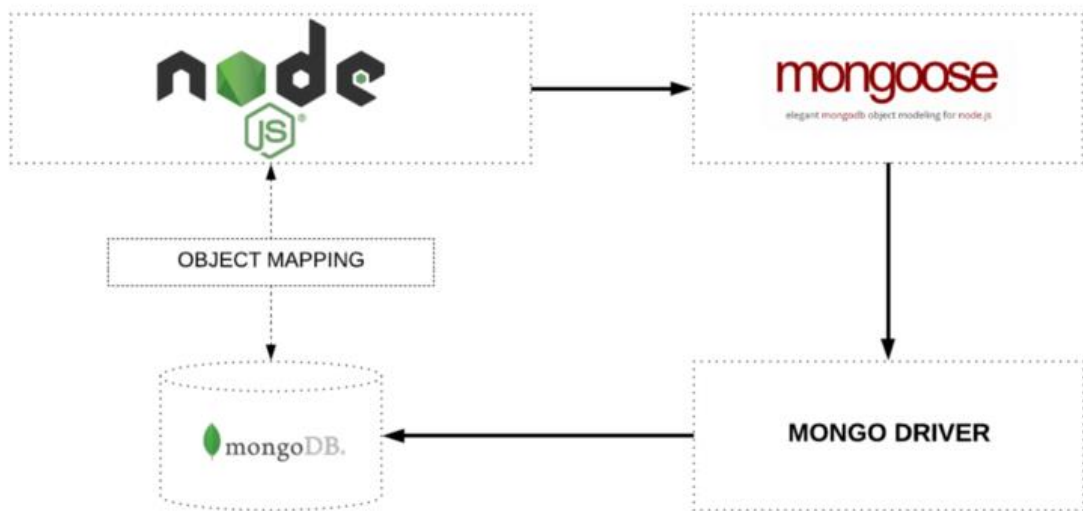


Рисунок 3.43 - Візуалізація роботи Node та Mongo через Mongoose

«Схема» Mongoose — це структура даних документа (або форма документа), яка виконується через прикладний рівень.

«Моделі» — це конструктори вищого порядку, які беруть схему та створюють екземпляр документа, еквівалентний записам у реляційній базі даних.

Модель Mongoose є оболонкою схеми Mongoose. Схема Mongoose визначає структуру документа, значення за замовчуванням, валідатори тощо, тоді як модель Mongoose забезпечує інтерфейс до бази даних для створення, запитів, оновлення, видалення записів тощо. [62]

Для зручного використання тіла http-запитів було взято middleware «body-parser».

body-parser – middleware, яке дозволяє розбирати тіла вхідних запитів перед обробниками, доступними у властивості req.body.

Об'єкт bodyParser відкриває різні фабрики для створення проміжного програмного забезпечення. Усі проміжні програми заповнюють властивість req.body проаналізованим тілом, якщо заголовок запиту Content-Type збігається з опцією типу, або порожнім об'єктом ({}), якщо немає тіла для аналізу, Content-Type не відповідає або виникла помилка [63]

Спільне використання ресурсів між джерелами (далі CORS) — це механізм на основі HTTP-заголовків, який дозволяє серверу вказувати будь-які джерела (домен, схему або порт), окрім власного, з якого браузер має дозволити завантажувати ресурси. CORS також покладається на механізм, за допомогою якого веб-переглядачі надсилають «переддруківний» запит на сервер, на якому розміщено ресурс із перехресним джерелом, щоб перевірити, чи сервер дозволить фактичний запит. Під час попередньої перевірки браузер надсилає заголовки, які вказують метод HTTP, і заголовки, які використовуватимуться у фактичному запиті. [64]

Для роботи з CORS було обрано однойменний NodeJS пакет «cors».

cors — це пакет node.js для надання проміжного програмного забезпечення Connect/Express, яке можна використовувати для ввімкнення CORS із різними параметрами. [65]

Для роботи з сеансами в Nest було обрано бібліотеку express-session

HTTP-сеанси забезпечують спосіб зберігання інформації про користувача в кількох запитах, що особливо корисно для програм MVC.

Приклад використання express-session в Nest показаний на рисунку 3.44. [66]

```
import * as session from 'express-session';
// somewhere in your initialization file
app.use(
  session({
    secret: 'my-secret',
    resave: false,
    saveUninitialized: false,
  }),
);
```

Рисунок 3.44 – Приклад використання express-session в Nest

Для того, щоб зберігати сеанси в MongoDB було обрано пакет connect-mongo

connect-mongo - сховище сеансів MongoDB для Connect і Express, написане на Typescript

На рисунку 3.45 зображений приклад використання connect-mongo при створенні express-session [67]

```
import session from 'express-session'
import MongoStore from 'connect-mongo'

app.use(session({
  secret: 'foo',
  store: MongoStore.create(options)
}));
```

Рисунок 3.45 – Приклад використання connect-mongo при створенні express-session

Веб-токен JSON (JWT) — це відкритий стандарт (RFC 7519), який визначає компактний і самодостатній спосіб безпечної передачі інформації між сторонами як об'єкт JSON. Цю інформацію можна перевірити та довіряти їй, оскільки вона має цифровий підпис. JWT можна підписати за допомогою секрету (з алгоритмом HMAC) або пари відкритих/приватних ключів за допомогою RSA або ECDSA.

Хоча JWT можна зашифрувати, щоб також забезпечити секретність між сторонами, ми зосередимося на підписаних токенах. Підписані токени можуть підтвердити цілісність претензій, що містяться в ньому, тоді як зашифровані токени приховують ці претензії від інших сторін. Коли токени підписуються за допомогою пар відкритих/приватних ключів, підпис також засвідчує, що лише сторона, яка володіє закритим ключем, є тією, яка його підписала. [68]

Для аутентифікації користувачів було обрано пакети @nestjs/passport, passport та @nestjs/jwt, passport-jwt(для роботи з jwt).

Passport — найпопулярніша бібліотека автентифікації node.js, добре відома спільноті та успішно використовується в багатьох робочих програмах. Інтегрувати

цю бібліотеку з додатком Nest за допомогою модуля просто `@nestjs/passport`. На високому рівні Passport виконує ряд кроків, щоб:

Автентифікувати користувача, підтвердивши його «облікові дані» (наприклад, ім'я користувача/пароль, веб-токен JSON (JWT) або маркер ідентифікації від постачальника ідентифікаційної інформації)

Керувати автентифікованим станом (шляхом видачі портативного маркера, наприклад JWT, або створення експрес-сеансу)

Додати інформацію про автентифікованого користувача до Requestоб'єкта для подальшого використання в обробниках маршрутів

Passport має багату екосистему стратегій , які реалізують різні механізми автентифікації. Незважаючи на просту концепцію, набір стратегій Passport великий і представляє велику різноманітність. Passport абстрагує ці різноманітні кроки в стандартний шаблон, а `@nestjs/passport` модуль обертає та стандартизує цей шаблон у знайомі конструкції Nest.

У цьому розділі ми реалізуємо повне рішення наскрізної автентифікації для сервера RESTful API за допомогою цих потужних і гнучких модулів. Ви можете використовувати концепції, описані тут, для реалізації будь-якої стратегії Passport для налаштування схеми автентифікації. Щоб побудувати цей повний приклад, виконайте дії, описані в цьому розділі.

Корисно думати про Passport як про міні-фреймворк сам по собі. Елегантність фреймворку полягає в тому, що він абстрагує процес автентифікації на кілька основних кроків, які розробник налаштовує відповідно до стратегії, яку він реалізує.

Це як фреймворк, тому що розробник налаштовує його, надаючи параметри налаштування (як звичайні об'єкти JSON) і спеціальний код у формі функцій зворотного виклику, які Passport викликає у відповідний час. Модуль `@nestjs/passport` обгортає цю структуру пакетом у стилі Nest, що полегшує інтеграцію в програму Nest.

Пакет `@nestjs/jwt` — це пакет утиліт, який допомагає маніпулювати JWT.

Пакет passport-jwt— це пакет Passport, який реалізує стратегію JWT і @types/passport-jwt надає визначення типів TypeScript. [69]

Для валідації(або перевірки) даних було обрано пакети class-validator та class-transformer.

Найкращою практикою є перевірка правильності будь-яких даних, надісланих у веб-програму. Для автоматичної перевірки вхідних запитів Nest надає кілька каналів, доступних прямо з коробки, наприклад, ValidationPipe

ValidationPipe дозволяє використовувати потужний пакет class-validator та його декоратори декларативної перевірки. ValidationPipe надає зручний підхід до застосування правил перевірки для всіх вхідних корисних даних клієнта, де конкретні правила оголошуються з простими анотаціями в оголошеннях локального класу/DTO в кожному модулі.

Для роботи з ValidationPipe потрібно встановити пакети class-validator та class-transformer. [70]

class-validator – пакет, який дозволяє використовувати перевірку на основі декоратора та не на основі декоратора. Внутрішньо використовує validator.js для виконання перевірки.

class-validator працює як у браузері, так і на платформах node.js. [71]

class-transformer - дозволяє перетворювати звичайний об'єкт на деякий екземпляр класу і навпаки. Також це дозволяє серіалізувати/десеріалізувати об'єкт на основі критеріїв. Цей інструмент надзвичайно корисний як для інтерфейсу, так і для серверу. [72]

Всього серверна частина підтримує 5 типів запитів.

Перший тип це запити на авторизацію та вихід з аккаунту:

- POST запит на “/authorization” - для авторизації;
- POST запит на “/authorization/logout” – для виходу.

Другий тип це запити на категорії:

- GET запит на “/category/all” – Щоб отримати всі категорії користувача;
- POST запит на “/category/add” – Щоб створити нову категорію;

- DELETE запит на “/category/:id” – Щоб видалити категорію;
- PUT запит на “/category/reorder” – Для зміни порядку категорії;
- PUT запит на “/category/:id” – Для зміни інформації категорії.

Третій тип це запити на ініціалізацію:

- GET запит на “/initialize” - отримання інформації про те, авторизований користувач чи ні.

Четвертий тип це запити на колір:

- GET запит на “/color/all” – Отримати всі кольори;
- POST запит на “/color/initialize” – Ініціалізувати кольори застосунку за допомогою окремого логіну та пароллю.

П’ятий тип це запити на статистику:

- GET запит на “/statistic/all” – Отримання всіх даних статистики користувача;
- GET запит на “/statistic/:id” – Отримання даних статистичного запису за його унікальним ідентифікатором;
- POST запит на “/statistic/add” – Створення нового статистичного запису;
- DELETE запит на “/statistic/:id” – Видалення статистичного запису за його унікальним ідентифікатором;
- PUT запит на “/statistic/:id” – Оновлення статистичного запису за його унікальним ідентифікатором.

Для приведення коду до єдиних стандартів і всьому застосунку було обрано Prettier.

Prettier — це самовпевнений форматувальник коду який бере код і передруковує його з нуля, враховуючи задані стилі.

Prettier забезпечує узгоджений стиль коду (тобто форматування коду, яке не впливає на абстрактне синтаксичне дерево) у всій кодовій базі, оскільки ігнорує оригінальний стиль, аналізуючи його та повторно друкуючи проаналізоване абстрактне синтаксичне дерево із власними правилами.

Безумовно, головна причина використання Prettier – не витратити час на форматування, тим самим пришвидшити написання проєкту та зберегти якість.

Prettier - єдиний «гід зі стилю», який працює повністю автоматично. [73]

Для відслідковування синтаксичних та помилок типів в коді при написанні застосунку був обраний ESLint

ESLint — це проект із відкритим вихідним кодом, який допомагає знаходити та виправляти проблеми з кодом JavaScript. ESLint статично аналізує код, щоб швидко знаходити проблеми.

Багато проблем, які ESLint знаходить, виправляються автоматично. Виправлення ESLint враховують синтаксис, тому у не виникає помилок, які виникають через традиційні алгоритми пошуку та заміни.

ESLint дозволяє попередньо обробляти код, використовувати спеціальні аналізатори та створювати власні правила, які працюють разом із вбудованими правилами ESLint. Налаштовувати ESLint так, щоб він працював саме так, як це потрібно для проекту. [74]

4 МЕТОДИКА РОБОТИ КОРИСТУВАЧА З СИСТЕМОЮ

4.1 Керівництво програміста

4.1.1 Призначення і умови використання програми

Призначенням цього сервісу є спрощення роботи з статистичними даними, тобто можливість зручно маніпулювати ними, як для користувачів які будуть використовувати його за допомогою клієнтської версії, так і для інших програмістів, які зможуть використовувати API для роботи з сервісом.

Основною умовою для використання локальних версій сервісу є встановлена NodeJS та встановлені залежності сервісу.

Основною умовою використання та необхідна складова для коректної роботи клієнтської частини, локальної версії, сервісу необхідно мати браузер, який має мінімальну версію не нижче 2020 року. Найбільш підходящими для використання є такі браузери, як Google Chrome, Safari, Mozilla Firefox, Opera та Microsoft Edge. А також мати встановлені залежності та встановлену локальну серверну частину сервісу.

Основною умовою використання серверної частини додатку, локальної версії, є наявність встановленої NodeJS з залежностями проекту та встановленої локально MongoDB.

4.1.2 Характеристики програми

Розмір вихідних файлів серверної частини – 1 мегабайт

Розмір вихідних файлів клієнтської частини – 1.4 мегабайт

Розмір підсумкових файлів серверної частини – 406 кілобайт

Розмір підсумкових файлів клієнтської частини(приблизний розмір передачі)
– 318 кілобайт

Нижче наведено функції, які включає програма:

- авторизація за допомогою облікового запису Google для зручного входу в систему без необхідності створювати новий обліковий запис та запам'ятовування логіну та паролю;
- можливість створювати, редагувати, видаляти та змінювати порядок категорій;
- користувачі повинні мати змогу переглядати інформацію про категорії в зручному табличному форматі;
- можливість створювати, редагувати, видаляти та змінювати порядок груп категорій. Одна категорія може належати до декількох груп;
- можливість створювати, редагувати, видаляти та змінювати порядок підкатегорій. Підкатегорії повинні значно полегшити статистичний облік даних користувачам;
- можливість створювати, редагувати та видаляти записи статистики в системі на основі категорій. Ця функція дозволяє зберігати важливі дані та інформацію;
- можливість перегляду активних сеансів користувача та їх завершення в разі необхідності;
- візуальне відображення статистичних даних у вигляді графіка з різноманітними параметрами групування та фільтрації;
- можливість переглядати записи статистики у вигляді таблиці з фільтраціями та сортуваннями за різними критеріями;
- можливість безпечного виходу з облікового запису без втрати даних;
- забезпечення безпеки даних. Забезпечення цілісності та безпеки даних є найважливішою функцією, яка гарантує, що інформація користувачів буде захищена від несанкціонованого доступу;

4.1.3 Звертання до програми

Для внесення змін в сервіс потрібно зробити клонування серверного та клієнтського репозиторіїв та встановити залежності.

Для серверної частини сервісу можливо зробити локальну копію віддаленої бази за допомогою вбудованого скрипту, який знаходиться в папці scripts

Для роботи з https можливо потрібно замінити само-підписані сертифікати які лежать в папках certs обох проєктів

Для запуску різних станів проєктів використовуються скрипти npm які записані в файлі package.json.

Для запуску проєктів за стандартом використовується `npm run start`

4.1.4 Вхідні і вихідні дані

Вхідними даними сервісу є:

- дані користувача для авторизації його в системі;
- категорії користувача;
- статичні дані.

Вихідними даними сервісу є:

- графічне відображення статистичних даних;
- відображення категорій;
- надання форм для створення статистичних даних та створення категорій.

4.2 Керівництво оператора

4.2.1 Призначення і умови використання програми

Призначенням цього сервісу є спрощення роботи з статистичними даними, тобто можливість зручно маніпулювати ними для користувачів які будуть використовувати його за допомогою клієнтської версії.

Основною умовою використання серверної частини додатку, віддаленої версії, є наявність середовища, яке може посилати HTTP(S) запити на сервер встановлюючи необхідні дані для отримання відповідей

Основною умовою використання та необхідна складова для коректної роботи клієнтської частини, віддаленої версії, сервісу необхідно мати браузер, який має мінімальну версію не нижче 2020 року. Найбільш підходящими для використання є такі браузери, як Google Chrome, Safari, Mozilla Firefox, Opera та Microsoft Edge.

4.2.2 Виконання програми

Клієнтська частина сервісу має 3 основні сторінки та 1 додаткову

В шапці клієнтської частини сервісу знаходиться логотип при натисненні на який користувач перейде на головну сторінку, меню, а також інформація про користувача, а саме його Ім'я та Прізвище, а також аватар.

На рисунку 4.1 продемонстрована шапка сервісу



Рисунок 4.1 – Демонстрація шапки сервісу

Якщо натиснути на інформацію про користувача з'явиться меню, в якому на даний момент можна вийти з аккаунту

На рисунку 4.2 продемонстровано меню користувача

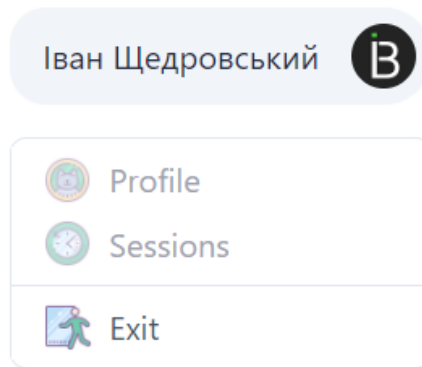


Рисунок 4.2 – Демонстрація меню користувача

Футор, або ж нижня панель сервісу має інформацію про поточну версію застосунку, посилання на [github](#) розробника, а також посилання на [tailwindcss](#) та [icons8](#), сервіси які були використані при розробці, та які по ліцензійній угоді потрібно указувати. На рисунку 4.3 продемонстрований футер сервісу

v1.1.0 next

Created by Itlaitoff



Рисунок 4.3 – Демонстрація футора сервісу

Домашня сторінка представляє собою форму для вводу нових даних статистики обираючи раніше створені категорії, вводячи коментар. Також для форми є додаткові опції, які можна побачити натиснувши на кнопку “Additional”. На рисунку 4.4 продемонстрована головна сторінка застосунку

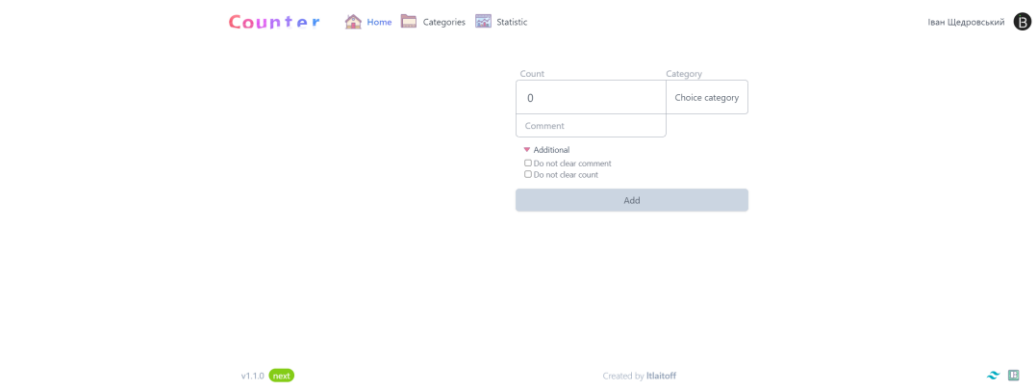


Рисунок 4.4 – Демонстрація головної сторінки застосунку

При натисненні на кнопку «Choice category» з’являється форма в якій можна вибрати категорію, представлена на рисунку 4.5

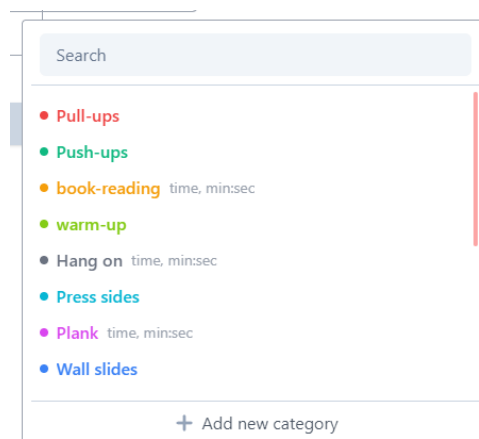


Рисунок 4.5 – Форма для вибору категорій

В формі вибору категорій реалізований функціонал пошуку за назвою, який продемонстрований на рисунку 4.6.

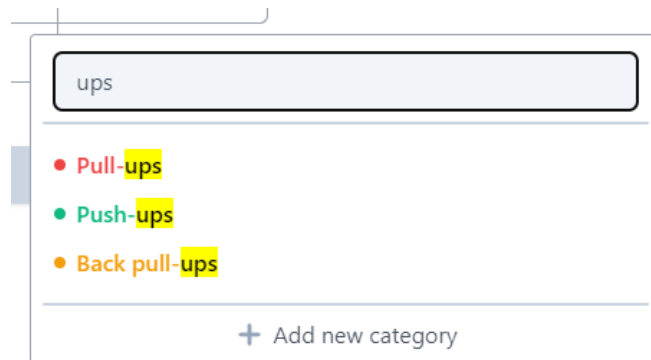


Рисунок 4.6 – Демонстрація пошуку категорії за назвою в формі вибору категорій

Також в формі вибору категорій реалізований функціонал додавання нової категорії за допомогою кнопки «Add new category» яка відкриє форму для додавання нової категорії

Сторінка категорій представляє собою дві кнопки та таблицю з категоріями користувача.

Перша кнопка «Add new category» при натисненні відкриває форму для додавання нової категорії. На рисунку 4.7 продемонстрована форма додавання нової категорії

Рисунок 4.7 – Демонстрація форми для додавання нової категорії

При натисненні на другу кнопку відбудеться примусове перезавантаження даних з API без перезавантаження сторінки. А також ця кнопка показує поточний статус категорій, всього їх 4: Не синхронізовано, синхронізація, помилка та синхронізовано.

Таблиця має в собі 4 стовпця:

- перший стовпець відповідає за колір категорії;
- другий за її назву;
- третій за коментар;
- четвертий за тип величин для зручності користувачу.

На рисунку 4.8 зображена вся сторінка категорій

	Name	Comment	Dimension
Red dot	Pull-ups	1 in 1	
Green dot	Push-ups	10 in 1	
Orange dot	book-reading		time, min:sec
Light green dot	warm-up		
Grey dot	Hang on	10s to 1. min 30s	time, min:sec
Teal dot	Press sides		
Purple dot	Plank	15s in 1	time, min:sec
Blue dot	Wall slides	Pull ups count	
Pink dot	Gluteal bridge		
Dark green dot	Iliac muscle stretch		time, min:sec
Pink dot	Gluteal bridge one leg		
Dark green dot	Squats	Squats count	
Dark green dot	Quadratus dorsi stretch		time, min:sec
Orange dot	Back pull-ups		
Light green dot	Brush	2kg+	
Grey dot	test		

Рисунок 4.8 – Демонстрація всієї сторінки категорій

При наведенні на рядок таблиці поряд з ним з'являється кнопка завдяки якій користувач може змінювати порядок категорії перетаскуванням, змінювати дані категорії або ж видалити її.

При наведенні курсору миші на цю кнопку користувач побачить підказку, в якій говориться, що потрібно зажати мишу та потягнути для зміни порядку, або ж клацнути для відкриття меню

На рисунку 4.9 представлена підказка для користувача для рядка таблиці категорій

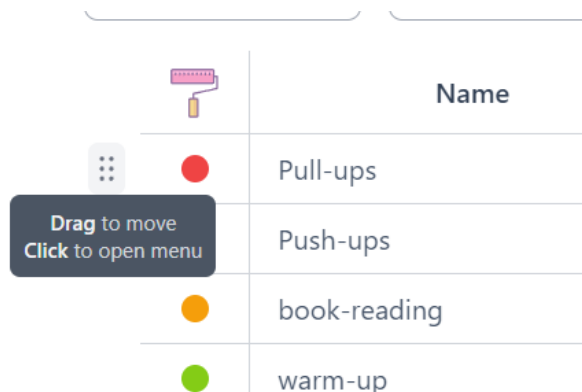


Рисунок 4.9 – Підказка для користувача для рядка таблиці категорій

При зміні порядку категорій, категорія яка переміщується замальована сірим кольором. На рисунку 4.10 представлена категорія, якій змінюють порядок







	Name	
	Push-ups	
	book-reading	
	Pull-ups	
	warm-up	
	Hang on	

Рисунок 4.10 – Категорія, якій змінюють порядок

При натисканні з'являється меню в якому користувач може змінити категорію, або ж видалити її.

На рисунку 4.11 зображено меню для зміни та видалення категорії

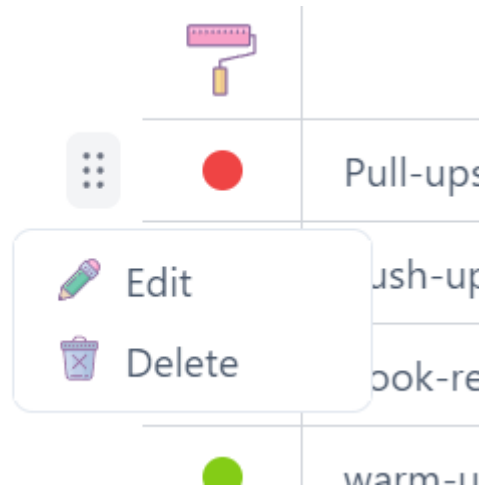


Рисунок 4.11 – Меню для зміни та видалення категорії

При натисканні на пункт меню «Edit» відкриється форма в якій можна змінити інформацію про категорію.

На рисунку 4.12 зображена форма для зміни інформації категорії

The image shows a web interface for editing a category. An 'Edit' modal is open, displaying the following fields:

- Name:** Pull-ups
- Comment:** 1 in 1
- Dimension:** (empty field)
- Color Selection:** A row of eight colored circles (red, pink, blue, teal, green, light green, grey, orange).
- Save:** A large blue button at the bottom of the modal.

In the background, a list of categories is visible, including 'Pull-ups' (red dot) and 'Iliac muscle stretch' (green dot).

Рисунок 4.12 – Форма для зміни інформації про категорію

На сторінці статистики зображений графік та таблиця даних. На рисунку 4.13 представлений приклад графіку статистики на сторінці статистики

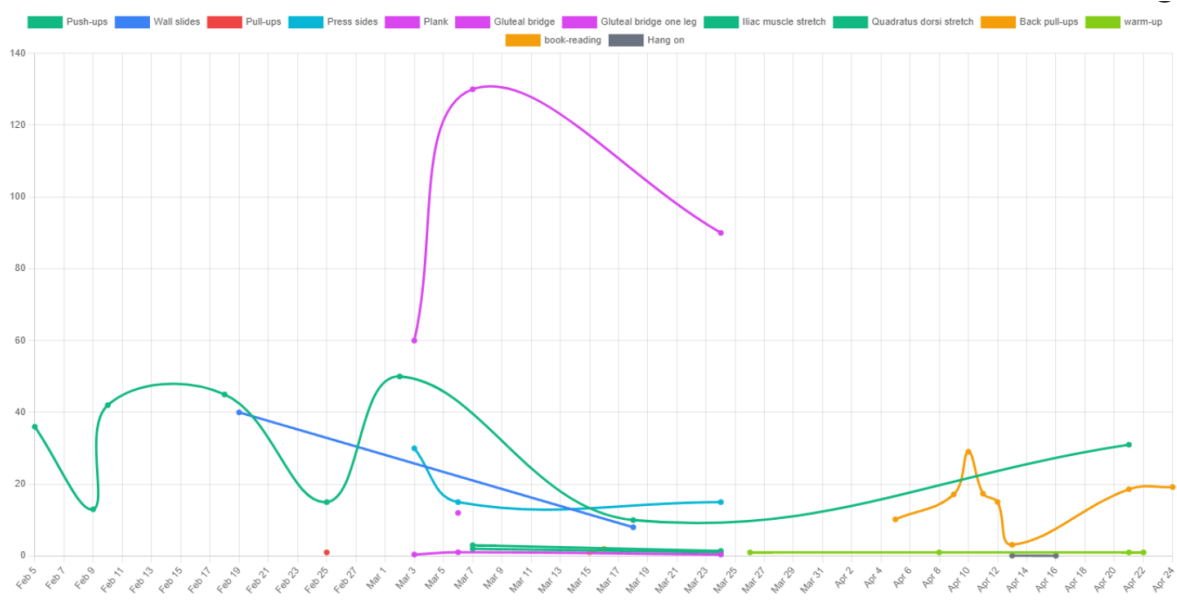


Рисунок 4.13 – Приклад графіку статистики

При наведенні на рядок таблиці статистики з'являються дві кнопки. За допомогою кнопки яка позначена як корзина можна видалити запис. За допомогою кнопки, яка позначена як ручка – можна змінити запис в відповідний форм. На рисунку 4.14 представлений приклад таблиці статистики з наведенням миші на рядок

Synchronized

Count	Comment	Category	Date
10		● Push-ups	Feb 5, 1:24:03 AM 2023
10		● Push-ups	Feb 5, 11:41:07 AM 2023
16		● Push-ups	Feb 5, 4:50:10 PM 2023
13		● Push-ups	Feb 9, 1:23:57 AM 2023
12		● Push-ups	Feb 10, 7:02:17 PM 2023
15		● Push-ups	Feb 10, 7:15:49 PM 2023
15		● Push-ups	Feb 10, 11:11:42 PM 2023
15		● Push-ups	Feb 18, 11:17:57 PM 2023
15		● Push-ups	Feb 18, 11:39:05 PM 2023
15		● Push-ups	Feb 18, 11:57:49 PM 2023
10		● Wall slides	Feb 19, 11:36:47 PM 2023
15		● Wall slides	Feb 19, 11:54:12 PM 2023
15		● Wall slides	Feb 19, 11:59:07 PM 2023
1		● Pull-ups	Feb 25, 1:55:43 PM 2023
15		● Push-ups	Feb 25, 1:58:16 PM 2023
10	1/10	● Push-ups	Mar 2, 10:44:53 PM 2023
10	2/10	● Push-ups	Mar 2, 10:50:02 PM 2023

Рисунок 4.14 – Приклад таблиці статистики з наведенням миші на рядок таблиці

При натисканні на кнопку позначену ручкою буде викликана форма для редагування даних запису статистики яка показана на рисунку 4.15

Count	Comment
10	Push-ups

Count

10

Comment

Date

02/05/2023 01:24:03 AM

Category:

Push-ups

Save

Рисунок 4.15 – Форма редагування даних статистики

На сторінці авторизація представлена одна кнопка для входу в аккаунт за допомогою Google Identity

На рисунку 4.16 представлений вигляд сторінки авторизації

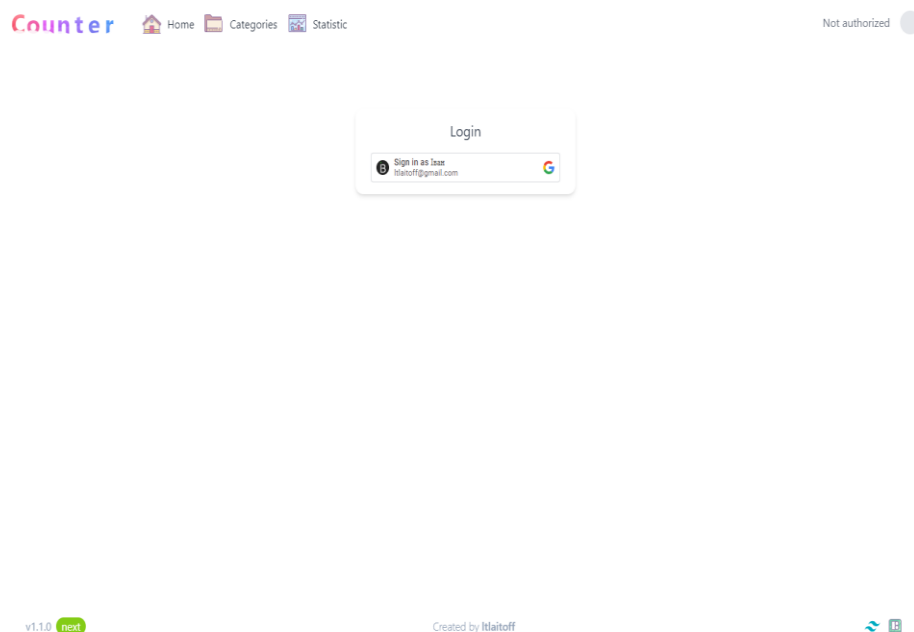


Рисунок 4.16 – Вигляд сторінки авторизації

5 ОРГАНІЗАЦІЙНО – ЕКОНОМІЧНИЙ РОЗДІЛ

5.1 Планування розробки програмного продукту

При плануванні розробки програмного продукту необхідно врахувати витрати робочого часу на виконання основних етапів роботи, що включають планування та аналіз вимог, що стосуються обраної предметної області, проєктування та тестування програмного продукту.

На рисунку 5.1 наданий укрупнений алгоритм розробки нової програми [75].



Рисунок 5.1 - Укрупнений алгоритм розробки програмного продукту

На підставі алгоритму визначаються етапи робіт, необхідні при розробці програмного продукту, їх трудомісткість, що буде вимірюватись в людино-днях. Таким чином можна визначити тривалість виконання роботи. Також процес

планування розробки програмного продукту буде включати визначення головних виконавців розробки додатку.

Результати планування заносяться до таблиці 5.1.

Таблиця 5.1 – Характеристика робіт з розробки програмного продукту

Найменування робіт	Трудомісткість		Виконавці		Тривалість розробки, днів
	люд.-днів	% до підсумку	спеціальність	кількість, осіб	
1. Вивчення та аналіз предметної області	10	28,57	Програміст, консультант	2	5
2. Розрахунки, розробка структури даних	6	17,14	Програміст, консультант	2	5
3. Розробка програми					
3.1Проектування програмного продукту	3	8,57	Програміст	1	3
3.2 Розробка модулів	7	20,0	Програміст	1	7
3.3 Компановка програми	5	14,29	Програміст	1	4
4.Тестування та відладка	6	17,14	Програміст, консультант	2	3
Разом	37	100	Програміст консультант	-	27

5.2 Розрахунок витрат на розробку програмного продукту

Складання кошторису витрат на розробку проєкту є складовою частиною процесу планування, у ході якого поряд із встановленням тимчасових характеристик розраховується потреба в грошових ресурсах. Витрати на розробку, поряд з іншими можливими витратами, входять у загальну величину капітальних вкладень в інвестиційний проєкт. Вони ж визначають ціну розробленого продукту при його реалізації, входячи в неї відразу або вроздріб, або у вигляді амортизаційних відрахувань [76].

5.2.1 Складання кошторису витрат на розробку

Витрати на розробку програмного продукту, грн.

$$C_{\text{разр}} = C_{\text{мат}} + C_{\text{ФОП}} + C_{\text{ЄСВ}} + C_{\text{Ам}} + C_{\text{ін}}, \quad (5.1)$$

де $C_{\text{мат}}$ – витрати матеріальні на поточний ремонт та електроенергію;

$C_{\text{ФОП}}$ – витрати на оплату праці;

$C_{\text{ЄСВ}}$ – відрахування єдиного соціального внеску;

$C_{\text{Ам}}$ – амортизаційні відрахування;

$C_{\text{ін}}$ – інші витрати [77].

Витрати на електроенергію, що була спожита під час написання та оформлення програмного продукту, грн.

$$B_{\text{ел.}} = N_{\text{н}} \cdot F_{\text{еф}} \cdot K_{\text{зав}} \cdot K_{\text{зп}} \cdot Ц_{\text{ел}}, \quad (5.2)$$

де N_n – номінальна потужність комп'ютера кВт/год., приймається за паспортом комп'ютера 0,4;

$F_{\text{еф}}$ – річний ефективний фонд роботи комп'ютера, год;

$K_{\text{зав.}}$ – середній коефіцієнт завантаження за часом, приймається 0,7;

$K_{\text{зп}}$ – коефіцієнт завантаження за потужністю, приймається 0,4;

$\text{Ц}_{\text{ел}}$ – ціна електроенергії, грн./кВт-год.

Для визначення ціни електроенергії приймається роздрібний тариф, який формується додаванням тарифу на послуги з розподілу електричної енергії для другого класу напруги та тарифу на послуги з передачі електричної енергії для непобутових споживачів відповідно до законів України «Про Національну комісію, що здійснює державне регулювання у сферах енергетики та комунальних послуг», «Про ринок електричної енергії», Ліцензійних умов провадження господарської діяльності з розподілу електричної енергії та Порядку встановлення (формування) тарифу на послуги з передачі електричної енергії. Актуальні тарифи без врахування ПДВ надаються на офіційному ресурсі Національної комісії, що здійснює державне регулювання у сферах енергетики та комунальних послуг. При розрахунку обов'язково включається в ціну електроенергії ПДВ (20%).

$$F_{\text{еф}} = D_{\text{роб}} \cdot t_{\text{зм}} - t_{\text{псв}} - t_{\text{обсл}}, \quad (5.3)$$

де $D_{\text{роб}}$ – робочі дні;

$t_{\text{зм}}$ – тривалість зміни за даними підприємства; приймається 8 год;

$t_{\text{псв}}$ – кількість годин скорочення тривалості робочого часу напередодні святкових і неробочих днів (ст.53 КЗпП), год.;

$t_{\text{обсл}}$ – час на технічне обслуговування ПК. Приймається 10 години.

$$F_{\text{еф}} = 26 \cdot 8 - 0 - 10 = 198,0$$

$$V_{\text{ел,річ}} = 0,4 \cdot 198 \cdot 0,7 \cdot 0,4 \cdot (1,51 + 5,82) = 162,55$$

Витрати на поточний ремонт устаткування, грн.

$$V_{\text{пот.рем.}} = V_{\text{перв.}} \cdot K_{\text{пот.рем.}}, \quad (5.4)$$

де $K_{\text{пот.рем.}}$ – коефіцієнт витрат на поточний ремонт, приймається 0,03.

$V_{\text{перв.}}$ – первісна вартість комп'ютера, грн.

Первісна вартість комп'ютера, грн.

$$V_{\text{перв.}} = V_{\text{придб.}} \cdot (1 + K_{\text{тр.м.}}), \quad (5.5)$$

де $V_{\text{придб.}}$ – вартість придбання (сума, сплачена постачальнику), грн.,
приймається за поточними ринковими цінами;

$K_{\text{тр.м.}}$ – коефіцієнт транспортних витрат, який враховує витрати на
доставку та налагодження, приймається 0,05.

$$V_{\text{перв.}} = 29\,000,00 \cdot (1 + 0,05) = 30\,450,00$$

$$V_{\text{пот.рем}} = 30\,450,00 \cdot 0,03 = 913,50$$

Проводимо розрахунок фонду оплати праці.

Фонд оплати праці складається з основної та додаткової заробітної плати програміста та консультанта.

Основна заробітна плата працівника визначається тарифними ставками, посадовими окладами, відрядними розцінками, її розмір залежить від результатів роботи самого працівника [78].

Фонд оплати праці фахівців, грн.

$$\text{ФОП} = \text{ЗП}_{\text{осн.}} + \text{ЗП}_{\text{дод.}}, \quad (5.6)$$

де $\text{ЗП}_{\text{осн.}}$, $\text{ЗП}_{\text{дод.}}$ – основна та додаткова заробітна плата фахівців - розроблювачів програмного продукту, грн.

Основна заробітна плата, грн.

$$\text{ЗП}_{\text{осн.}} = \text{ЗП}_{\text{ден.}} \cdot \text{Д}_{\text{розр}}, \quad (5.7)$$

де $\text{ЗП}_{\text{ден.}}$ – денна заробітна плата фахівця, грн./день;

$\text{Д}_{\text{розр.}}$ – тривалість розробки програмного продукту, днів.

Денна заробітна плата фахівців, грн./день

$$\text{ЗП}_{\text{ден}} = \frac{\text{О}}{\text{Д}_{\text{роб}}}, \quad (5.8)$$

де О – посадовий оклад фахівця, грн.

$\text{Д}_{\text{роб}}$ – кількість робочих днів в місяці, приймаємо 22.

Для програміста приймається посадовий оклад, визначений як добуток п'ятикратної мінімальної заробітної плати, встановленої Законом України «Про державний бюджет України» на поточний рік на момент виконання завдання, та коефіцієнтом диференціації, що утворюється додаванням до одиниці номеру студента за списком у журналі, поділеного на 100.

Для консультанта прийняти посадовий оклад, що буде дорівнювати потрійній мінімальній заробітній платі, встановленій чинним законодавством України на момент виконання завдання.

$$ЗП_{\text{ден.прогр}} = \frac{35\,845}{22} = 1\,629,32$$

$$ЗП_{\text{ден.конс}} = \frac{20\,100}{22} = 913,64$$

$$ЗП_{\text{осн. прогр.}} = 1\,629,32 \cdot 26 = 42\,362,32$$

$$ЗП_{\text{осн. конс.}} = 913,64 \cdot 11 = 10\,050,04$$

Результати розрахунків зводяться до таблиці 5.2.

Таблиця 5.2 – Розрахунок основної заробітної плати фахівців

Посада виконавця	Чисельність фахівців, осіб	Місячний оклад фахівців, грн.	Сума основної заробітної плати фахівців, грн.
Програміст	1	35 845,00	42 362,32
Консультант	1	20 100,00	10 050,04
Разом	2	-	52 412,36

Величина додаткової заробітної плати визначається кінцевими результатами діяльності підприємства і виступає у формі премій, винагород, заохочувальних виплат, доплат за особливі умови праці, оплати за невідпрацьований час (відпустки, лікарняні).

Додаткова заробітна плата фахівців, грн.

$$ЗП_{\text{дод.}} = ЗП_{\text{осн.}} \cdot K_{\text{дод.зп.}}, \quad (5.9)$$

де $K_{\text{дод.зп.}}$ - коефіцієнт додаткової заробітної плати, приймається 0,2.

$$ЗП_{\text{дод. прогр.}} = 42\,362,32 \cdot 0,2 = 8\,472,46$$

$$ЗП_{\text{дод. конс.}} = 10\,050,04 \cdot 0,2 = 2\,010,01$$

Загальна заробітна плата фахівців, грн.

$$\text{ФОП}_{\text{прогр.}} = 42\,362,32 + 8\,472,46 = 50\,834,78$$

$$\text{ФОП}_{\text{конс.}} = 10\,050,04 + 2\,010,01 = 12\,060,05$$

Результати розрахунків фонду заробітної плати наводяться у таблиці 5.3.

Таблиця 5.3 – Фонд заробітної плати фахівців

Посада виконавця	Основна зарплата, грн.	Додаткова зарплата, грн.	Всього фонд оплати праці, грн.
Програміст	42 362,32	8 472,46	50 834,78
Консультант	10 050,04	2 010,01	12 060,05
Разом	52 412,36	10 482,47	62 894,83

Єдиний соціальний внесок (ЄСВ) – обов'язковий платіж до системи загальнообов'язкового державного соціального страхування, що справляється в Україні з метою забезпечення страхових виплат за поточними видами загальнообов'язкового державного соціального страхування [79].

Розмір ЄСВ, грн.

$$B_{\text{есв}} = (3\Pi_{\text{осн.}} + 3\Pi_{\text{дод.}}) \cdot K_{\text{есв}}, \quad (5.10)$$

де $K_{\text{есв}}$ – коефіцієнт відрахувань єдиного соціального внеску, прийняти згідно діючого законодавства.

$$B_{\text{есв.}} = 62\,894,83 \cdot 0,22 = 13\,836,86$$

Місячна сума амортизаційних відрахувань, грн.

$$A_{\text{міс}} = \frac{B_{\text{перв.}} \cdot H_a}{100 \cdot 12}, \quad (5.11)$$

де $B_{\text{перв.}}$ – первісна вартість комп'ютера, грн.;

H_a – річна норма амортизації, % (згідно Податкового кодексу України мінімальний термін експлуатації комп'ютера 2 роки, отже, річна норма амортизації приймається 50%).

$$A_{\text{міс}} = \frac{30\,450 \cdot 0,5}{100 \cdot 12} = 1\,268,75$$

Інші витрати на утримання та експлуатацію обладнання, грн.

$$B_{\text{ін.}} = (B_{\text{пот.рем}} + B_{\text{ел.}} + \text{ФОП} + B_{\text{есв}} + A_{\text{міс}}) \cdot K_{\text{інш.}}, \quad (5.12)$$

де $K_{\text{інш.}}$ – коефіцієнт інших витрат, приймається 0,03

$$B_{\text{ін.}} = (913,50 + 162,55 + 62\,894,83 + 13\,836,86 + 1\,268,75) \cdot 0,03 = 2\,372,29$$

Річні витрати на утримання та експлуатацію обладнання (спеціального устаткування) зведено до таблиці 5.4.

Таблиця 5.4 – Кошторис витрат на утримання і експлуатацію обладнання

Найменування статей витрат	Сума, грн.
1 Матеріальні витрати (витрати на поточний ремонт та електроенергію)	1 076,05
2 Витрати на оплату праці	62 894,83
3 Відрахування єдиного соціального внеску	13 836,86
4 Амортизаційні відрахування	1 268,75
5 Інші витрати	2 372,29
Разом	81 448,78

5.2.2 Розрахунок собівартості програмного продукту

Витрати, пов'язані з виробництвом і реалізацією одного програмного продукту, грн.

$$C_{c/v} = C_{\text{мат}} + C_{\text{осн}} + C_{\text{дод}} + C_{\text{есв}} + C_{\text{уео}} + C_{\text{накл}}, \quad (5.13)$$

де $C_{\text{мат}}$ - витрати на основні матеріали, покупні напівфабрикати та комплектуючі, грн/шт;

$C_{\text{осн}}$ - основна заробітна плата, грн.;

$C_{\text{дод}}$ - додаткова заробітна плата, грн.;

$C_{\text{есв}}$ - відрахування єдиного соціального внеску, грн.;

$C_{\text{уео}}$ – витрати на утримання та експлуатацію обладнання, грн.;

$C_{\text{накл}}$ - накладні видатки, грн.

До складу матеріальних витрат включають вартість витратних матеріалів, що знадобляться при розробці та супроводі програмного продукту, оформлення технічної документації для здачі дипломної роботи [80].

Прямі матеріальні витрати, грн.

$$V_{\text{мат.}} = \sum q_i \cdot C_i, \quad (5.14)$$

де C_i – ціна i -го матеріалу, грн./од.;

q_i – норма витрат матеріалу, натуральних одиниць

Розрахунок прямих матеріальних витрат приведений у таблиці 5.5.

Таблиця 5.5 – Прямі матеріальні витрати

Назва матеріалу	Одиниця вимірювання	Норма витрат, нат.од.	Ціна, грн./нат.од.	Сума, грн.
1. Папір	пачка	1	250,00	250,00
2. Картридж	шт.	1	400,00	400,00
3. Флеш накопичувач 32 Гб	шт.	1	200,00	200,00
Разом	-	-	-	850, 00

Експлуатаційні витрати на 1 годину роботи комп'ютера, грн./год.,

$$V_{\text{експл.1год}} = \frac{B_{\text{уео}}}{F_{\text{еф}}}, \quad (5.15)$$

де V_{yco} – витрати на утримання та експлуатацію обладнання, грн.

$$V_{yco} = V_{\text{пот.рем}} + V_{\text{ел.}} + A_{\text{міс.}} \quad (5.16)$$

$$V_{yco} = 913,50 + 162,55 + 1\,268,75 = 2\,344,80$$

$$V_{\text{експл. 1 год}} = \frac{2\,344,80}{198,0} = 11,84$$

Експлуатаційні витрати на весь час роботи комп'ютера при написанні програмного продукту, грн.

$$V_{\text{експл.}} = V_{\text{експ. 1 год}} \cdot t_{\text{м.ч.}}, \quad (5.17)$$

де $t_{\text{м.ч.}}$ – час роботи ПК при написанні програмного продукту, в середньому робота за комп'ютером приймається 4 год./день

$$t_{\text{м.ч.}} = D_{\text{роб.прогр.}} \cdot 4, \quad (5.18)$$

де $D_{\text{роб.прогр.}}$ – кількість днів роботи програміста при розробці програмного продукту

$$t_{\text{м.ч.}} = 26 \cdot 4 = 104$$

$$V_{\text{експл.}} = 11,84 \cdot 104 = 1\,231,36$$

Накладні (непрямі) витрати – це витрати, які пов'язані не з виготовленням конкретних виробів, а з процесом виробництва в цілому (зарплата управлінського персоналу, утримання та експлуатація будівель, комунальні послуги та ін.), грн.

$$V_{\text{накл.}} = (ЗП_{\text{осн.прогр.}} + ЗП_{\text{осн.конс.}}) \cdot K_{\text{накл.}}, \quad (5.19)$$

де $K_{\text{накл.}}$ – коефіцієнт накладних витрат, приймається 0,50.

$$V_{\text{накл.}} = 52\,412,36 \cdot 0,5 = 26\,206,68$$

Результати розрахунків витрат на розробку програмного продукту наводяться в таблиці 5.6.

Таблиця 5.6 – Калькуляція собівартості робіт з розробки програмного продукту

Найменування статей витрат	Сума, грн.	Питома вага, %
1 Матеріальні витрати	850,00	0,81
2 Основна заробітна плата	52 412,36	49,91
3 Додаткова заробітна плата	10 482,47	9,98
4 Відрахування єдиного соціального внеску	13 836,86	13,18
5 Експлуатаційні витрати	1 231,36	1,17
6 Накладні витрати	26 206,68	24,95
Разом	105 019,73	100

5.3 Оцінка ефективності проєкту

В основі процесу прийняття будь-яких рішень інвестиційного характеру лежить оцінка та порівняння обсягу необхідних капітальних вкладень. Різниця вартісної оцінки результатів і витрат за час існування проєкту визначає величину економічного ефекту. Перевищення результатів над витратами представляє абсолютний ефект. Відношення його величини до зроблених витрат характеризують відносний ефект - ефективність капітальних вкладень, зворотня

величина - строк окупності капітальних вкладень.

Якщо при реалізації розробленого продукту мають місце разові первісні капітальні вкладення та картина грошових потоків за роками не змінюється, то оцінку економічної ефективності проекту можна проводити по одному році [80].

Для ухвалення остаточного рішення про можливість практичної реалізації проекту робиться комплексна оцінка наступних економічних показників.

Річний економічний ефект, грн.

$$E_p = C_{c/v} - E_n \cdot K, \quad (5.20)$$

де K – обсяг капітальних вкладень споживача програми, пов'язаних з її розробкою і впровадженням;

E_n – нормативний коефіцієнт економічної ефективності капітальних вкладень, який встановлюється Міністерством економіки України на певний період. Прийняти 0,15.

$$K = K_{пв} + K_{вф}, \quad (5.21)$$

де $K_{пв}$ – попередньо виробничі витрати, грн.;

$K_{вф}$ – капітальні вкладення у виробничі фонди, необхідні для впровадження програми, грн.

Попередньовиробничі витрати, грн.

$$K_{пв} = \frac{C_{c/v}}{n} + K_{пр.ін.}, \quad (5.22)$$

де n - кількість споживачів даної програми, прийняти не менше 2-х осіб;

$K_{пр.ін.}$ - інші попередньовиробничі витрати, грн.

Капітальні вкладення у виробничі фонди включають витрати на придбання, установку, монтаж і налагодження комплексу технічних засобів (КТЗ) з урахуванням можливого використання КТЗ як для рішення задач, для яких призначена розроблювальна програма, так і для задач, розв'язуваних за допомогою інших програм. Питомі капітальні вкладення у виробничі фонди, що приходяться на частку даного програмного продукту, грн.

$$K_{\text{вф}} = \frac{K_{\text{КТЗ}} \cdot T_{\text{КТЗ}}}{F_{\text{еф.КТЗ}}}, \quad (5.23)$$

де $K_{\text{КТЗ}}$ – витрати на придбання, установку, монтаж і налагодження КТЗ, грн.;

$T_{\text{КТЗ}}$ – машинний час КТЗ, потрібний споживачу програми для задач, розв'язуваних за допомогою даної програми, машино-годин/рік (прийняти 11 місяців);

$F_{\text{еф.КТЗ}}$ – річний ефективний фонд часу роботи КТЗ, машино-годин/рік

$$K_{\text{КТЗ}} = n \cdot B_{\text{перв}}, \quad (5.24)$$

де n – кількість місяців роботи над розробкою програмного продукту;

B_6 - балансова вартість комп'ютера

Розраховуємо річний економічний ефект.

$$K_{\text{ПВ}} = \frac{105\,019,73}{2} + 0 = 52\,509,87$$

$$K_{\text{КТЗ}} = 26/22 \cdot 30\,450,00 = 35\,986,36$$

$$K_{\text{вф}} = \frac{35\,986,36 \cdot 11}{198 \cdot 12} = 166,60$$

$$K = 35\,986,36 + 166,60 = 36\,152,96$$

$$E_p = 105\,019,73 - 36\,152,96 \cdot 0,15 = 99\,596,79$$

Проект можна рекомендувати, якщо величина E_p при заданому значенні E_n позитивна: $E_p > 0$.

Для рішення питання про доцільність інвестування крім річного економічного ефекту (E_p) як критерію оцінки проєктів також використовують строк окупності капітальних вкладень, $T_{ок}$, що вимірюється в роках

$$T_{ок} = \frac{K}{E_p}, \quad (5.25)$$

$$T_{ок} = \frac{36\,152,96}{99\,596,79} = 0,36$$

Гранична величина строку окупності обмежується тривалістю проєкту.

Створюваний додаток ефективний та доцільний для використання.

6 ОХОРОНА ПРАЦІ КОРИСТУВАЧІВ КОМП'ЮТЕРІВ

6.1 Правове забезпечення заходів щодо охорони праці користувачів комп'ютерів

Питаннями охорони праці в міжнародному масштабі та розробкою конвенцій, рекомендацій з різних соціально-правових проблем займається Міжнародна Організація праці (МОП). За роки діяльності МОП, підхід до проблеми охорони праці, який базувався на першочерговій увазі до випадків найбільш серйозних порушень і галузей з найвищим рівнем травматизму та захворюваності, значно розширився і переріс у всеосяжну систему, метою якої є досягнення найвищого рівня безпеки та гігієни праці у всіх галузях та професіях. Все вищесказане в повній мірі стосується користувачів комп'ютерів, з огляду на значний ріст цієї професії у всьому світі.

Першою країною, яка почала займатися створенням стандартів, що регламентують роботу з комп'ютерами, є Швеція. Там ще в першій половині 80-х років почалися серйозні дослідження умов праці з ВДТ та їх вплив на здоров'я користувачів. До цих досліджень було залучено більше 20 шведських наукових організацій, в тому числі Шведський інститут з питань захисту від випромінювань, Національна Рада з техніки безпеки і гігієни праці, Шведський національний комітет з вимірювань та випробовувань та ін.

В результаті проведеної роботи був розроблений стандарт МРК II 1990, який вважається базовим. Методика МРК II направлена, в основному, на перевірку двох характеристик відеотерміналів: візуальних ергономічних (яскравість, нелінійність, чіткість, колір, коефіцієнт відбиття, неортогональність та ін.) та емісійних (потужність дози рентгенівського випромінювання, напруженість електромагнітного поля за електричною та магнітною складовою в різних діапазонах частот, електростатичний потенціал). Пізніше з'явилися стандарти Шведської конфедерації профспілок TCO 92 та TCO 95 з ще більш жорсткими

вимогами стосовно характеристик відеотерміналів, зокрема неіонізуючого електромагнітного випромінювання

Окрім того, ТСО 95 регламентує характеристики ВДТ щодо енергозбереження та містить екологічні вимоги, у відповідності з якими в конструкціях пристроїв не повинні міститись галогенні пластмаси, фреони, що пов'язано з охороною озонowego шару планети. Пакувальні матеріали не повинні містити хлоридів та бромідів і підлягати вторинній нетоксичній переробці.

Недавно вийшов новий стандарт — ТСО 99, у якому питання ергономіки, енергозбереження та екології також перебувають на чільному місці.

На шведські стандарти спираються практично всі ведучі фірми-виробники відеотерміналів. [81]

Серед низки розроблених нормативних документів щодо використання ВДТ найбільш часто використовуються наступні стандарти:

На сьогодні найбільш новим нормативним документом, що регламентує роботу користувачів ВДТ, є ISO 9241, який складається з наступних частин:

- загальний вступ;
- poradnik z використання;
- вимоги для представлення даних;
- вимоги до клавіатури;
- вимоги до розміщення обладнання на робочому місці;
- вимоги до навколишнього середовища;
- вимоги до відблиску екрана;
- вимоги до кольорового зображення;
- вимоги до неклавіатурних пристроїв введення;
- принципи організації діалогу;
- poradnik z використання ВДТ;
- вимоги до представлення інформації;
- poradnik для користувача;
- діалогове меню;

- діалогові команди;
- управління діалогом;
- форма наповнення діалогів.

Основоположним законодавчим документом в галузі охорони праці є Закон України "Про охорону праці", дія якого поширюється на всі підприємства, установи і організації незалежно від форм власності та видів діяльності, на усіх громадян, які працюють, а також залучені до праці на цих підприємствах. Цей Закон визначає основні положення щодо реалізації конституційного права громадян про охорону їх життя і здоров'я в процесі трудової реальності, регулює за участю відповідних державних органів відносини між власниками підприємства, установи чи організації або уповноваженим ними органом і працівником з питань безпеки, гігієни праці та виробничого середовища і встановлює єдиний порядок організації ОП.

Основним нормативним документом щодо забезпечення охорони праці користувачів ВДТ є ДНАОП 0,00-1.31-99 "Правила охорони праці під час експлуатації електронно-обчислювальних машин". Дані Правила встановлюють вимоги безпеки та санітарно-гігієнічні вимоги : до обладнання робочих місць користувачів електронно-обчислювальних; машин і персональних комп'ютерів (дані ЕОМ) та працівників, що виконують обслуговування, ремонт та налагодження ЕОМ та роботи з застосуванням ЕОМ, відповідно до сучасного стану техніки та наукових досліджень у сфері безпечної організації робіт з експлуатації ЕОМ та з урахуванням положень міжнародних нормативно-правових актів з цих питань. [81]

ДНАОП 0.00-1.31-99 поширюється на всі підприємства, установи, організації, юридичні особи, крім зазначених нижче, незалежно від форми власності, відомчої належності, видів діяльності та на фізичних осіб , (що займаються підприємницькою діяльністю з правом найму робочої сили), які здійснюють розробку, виробництво та застосування ЕОМ, у тому числі й на тих, які мають робочі місця, обладнані ЕОМ, або виконують обслуговування, ремонт та налагодження ЕОМ.

ДНАОП 0.00-1.31-99 складається з таких розділів:

- загальні положення;
- вимоги до виробничих приміщень;
- вимоги до обладнання;
- вимоги до розміщення устаткування та організації робочих місць;
- вимоги безпеки під час експлуатації, обслуговування, ремонту та налагодження ЕОМ;
- режими праці та відпочинку;
- вимоги до виробничого персоналу;
- обов'язки, права та відповідальність за порушення Правил.

В цьому розділі сформульовані вимоги до виробничого персоналу. В ньому вказується, що усі працівники, які виконують роботи, пов'язані з експлуатацією, обслуговуванням, налагодженням та ремонтом ЕОМ, підлягають обов'язковому медичному огляду: попередньому — під час оформлення на роботу та періодичному — протягом трудової діяльності. [81]

Згідно з вимогами цього розділу Правил посадові особи та спеціалісти, інші працівники підприємств, які організовують та виконують роботи, пов'язані з експлуатацією, профілактичним обслуговуванням, налагодженням та ремонтом ЕОМ, проходять підготовку (підвищення кваліфікації), перевірку знань з охорони праці, даних Правил та питань пожежної безпеки, а також інструктажі в порядку, передбаченому Типовим положенням про навчання з питань охорони праці, затвердженим наказом Держнаглядохоронпраці 17.02.99 №27, Типовим положенням про спеціальне навчання, інструктажі та перевірку знань з питань пожежної безпеки на підприємствах, в установах та організаціях України і Переліком посад, при призначенні на які особи зобов'язані проходити навчання та перевірку знань з питань пожежної безпеки, та порядком її організації, затвердженими наказом МВС України від 17. 11. 94 № 628.

Працівники, що виконують роботи з профілактичного обслуговування, налагодження і ремонту ЕОМ при включеному живленні, та при інших роботах,

передбачених Переліком робіт з підвищеною небезпекою, затвердженим наказом Держнаглядохоронпраці, від 30.11.93 № 123, зобов'язані проходити попереднє спеціальне навчання та один раз на рік перевірку знань відповідних нормативних актів з охорони праці.

Наголошується на забороні допускати до роботи осіб, що в установленому порядку не пройшли навчання, інструктаж та перевірку знань з охорони праці та пожежної безпеки. Вказується, що до робіт з обслуговування, налагодження та ремонту ЕОМ допускаються особи, які мають кваліфікаційну групу з електробезпеки не нижче III.

У сьомому розділі Правил також підкреслюється, що забороняється допускати осіб, віком до 18 років, до самостійних робіт в електроустановках та на електрообладнанні під час профілактичного обслуговування, налагодження, ремонту ЕОМ та при інших роботах, передбачених Переліком важких, робіт та робіт зі шкідливими та небезпечними умовами праці, на яких забороняється застосовувати працю неповнолітніх, затвердженим наказом Міністерства охорони здоров'я України від 31.03.94 № 46. [82]

Згідно зі стат.10 Закону України "Про охорону праці" працівники, які виконують роботи з експлуатації, обслуговування, налагодження та ремонту ЕОМ, повинні забезпечуватись належними засобами індивідуального захисту відповідно до чинних норм.

У восьмому розділі ДНАОП 0.00-1.31-99 розглянуто питання щодо обов'язків, прав та відповідальності за порушення Правил як власника так і працівника.

Відповідно до Закону України "Про охорону праці" власник:

- на підставі цих Правил, інших нормативно-правових актів про охорону праці, примірних інструкцій, інструкцій з експлуатації обладнання розробляє та затверджує інструкції з охорони праці за професіями або на окремі види робіт з урахуванням фактичних умов проведення робіт, технології, наявності обладнання й інструменту, засобів захисту та рівня підготовки виконавців, проводить

відповідне навчання та інструктажі з працівниками;

- вживає необхідних заходів з тим, щоб робочі місця та засоби виробництва протягом всього часу їх використання підтримувались у справному та безпечному стані, а виявлені недоліки, що впливають на охорону праці та захист здоров'я працівників, були своєчасно усунуті;

- відповідно до Порядку проведення атестації робочих місць за умовами праці проводить атестацію робочих місць для оцінки умов праці. На підставі аналізу проведеної атестації вживає заходів для унеможливлення виникнення небезпечних та шкідливих факторів;

- організовує роботу працівника таким чином, щоб повсякденна робота з відеотерміналом регулярно переривалась паузами або іншими видами діяльності, що знижують навантаження, обумовлене роботою з відеотерміналом, відповідно до вимог Правил;

- організовує проведення обстеження зору працівника окулістом не за кошти працівника перед початком роботи з відеотерміналом, потім періодично відповідно до ДСанПіН 3.3.2-007-98, а також при виникненні скарг на погіршення зору безкоштовно надає індивідуальні окуляри коригування зору відповідно до умов роботи з відеотерміналом, якщо результати обстеження показали, що вони є необхідними;

- забезпечує даними Правилами підприємство, керівників служб та структурних підрозділів, безпосередніх керівників робіт, робочі місця яких обладнані відеотерміналами та ЕОМ, та/або які виконують обслуговування, ремонт та налагодження комп'ютерної техніки. [81]

Працівник має право:

- на відповідне дослідження очей та зору особою відповідної кваліфікації при виникненні скарг на погіршення зору, яке може бути наслідком роботи на відеотерміналі;

- на одержання за рахунок роботодавця індивідуальних засобів коригування зору відповідно до умов роботи за відеотерміналом, якщо результати досліджень

показали, що вони є необхідними;

- на інформацію про всі важливі питання його здоров'я та безпеки, пов'язані з перебуванням за робочим місцем, а також про заходи, що вживаються на виконання вимог цих Правил.

В цьому ж розділі наведено перелік обов'язків, які покладаються на працівника. Так, відповідно до Закону України "Про охорону праці" працівник зобов'язаний:

- знати та виконувати вимоги нормативно-правових актів про охорону праці, даних Правил, інструкцій з охорони праці, інструкцій щодо експлуатації застосовуваного обладнання, правила поведінки з устаткуванням, інструментом та іншими засобами виробництва;

- використовувати засоби колективного та індивідуального захисту;

- дотримуватись зобов'язань з охорони праці, передбачених колективним договором (угодою, трудовим договором) та правилами внутрішнього трудового розпорядку підприємства, проходити в установленому порядку попередні та періодичні медичні огляди;

- негайно повідомляти власника або безпосереднього керівника і робіт про кожну виявлену серйозну та безпосередню небезпеку, про будь-яке пошкодження захисних пристроїв та засобів захисту, про несправності устаткування, інструменту та інших засобів виробництва;

- не відключати захисні пристрої, не проводити самовільних змін конструкції і складу устаткування або його технічного налагоджування.

В кінці розділу сформульовані положення стосовно відповідальності за виконання даних Правил. Так нормативним документом визначено, що власники, керівники служб та структурних підрозділів, безпосередні керівники робіт та інші посадові особи підприємств, фізичні особи, що займаються підприємницькою діяльністю з правом найму робочої сили, працівники несуть відповідальність за виконання вимог даних Правил у межах покладених на них завдань та функціональних обов'язків згідно з чинним законодавством. В той же час за

безпеку експлуатації, обслуговування, ремонту та налагодження ЕОМ, а також за відповідність обладнання, виробничих приміщень, робочих місць даним Правилам відповідає власник. В нормативному документі наголошується, що особи, винні в порушенні цих Правил, несуть дисциплінарну, адміністративну, матеріальну або кримінальну відповідальність згідно з чинним законодавством. [82]

Окрім ДНАОП 0.00-1.31-99 найпильнішої уваги заслуговують Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин (ДСанПіН 3.3.2-007-98). Дані Санітарні правила складаються з наступних розділів:

- загальні положення.
- вимоги до виробничих приміщень для експлуатації ВДТ ЕОМ та ПЕОМ.
- гігієнічні вимоги до параметрів виробничого середовища приміщень з ВДТ ЕОМ та ПЕОМ.
- гігієнічні вимоги до організації і обладнання робочих місць з ВДТ ЕОМ та ПЕОМ.
- вимоги до режимів праці і відпочинку при роботі з ВДТ ЕОМ та ПЕОМ.
- вимоги до профілактичних медичних оглядів.

Зміст переважної більшості розділів ДСанПіН 3.3.2-007-98 в основному збігається зі змістом відповідних розділів ДНАОП 0.00-1.31-99. Тому зупинимось лише на вимогах до режимів праці і відпочинку, оскільки саме вони займають важливе місце в комплексі заходів щодо забезпечення безпеки праці та її високої продуктивності, профілактики перевтоми та відхилень у стані здоров'я користувачів ВДТ.

В Санітарних правилах вказано, що при організації праці, що пов'язана з використанням ВДТ ЕОМ і ПЕОМ слід передбачити внутрішньозмінні регламентовані перерви для відпочинку, які передують появі об'єктивних і суб'єктивних ознак втоми і зниження працездатності. Роз'яснюється, що при виконанні протягом дня робіт, які належать до різних видів трудової діяльності, за

основну роботу з ВДТ ЕОМ і ПЕОМ слід вважати такою, що займає не менше 50% часу впродовж робочої зміни чи робочого дня.

Протягом робочої зміни мають передбачатися:

- перерви для відпочинку і вживання їжі (обідні перерви);
- перерви для відпочинку і особистих потреб (згідно з трудовими нормами);
- додаткові перерви, що вводяться для окремих професій з урахуванням особливостей трудової діяльності.

В Санітарних правилах робиться висновок про те, що для зниження нервово-емоційного напруження, запобігання втомі зорового аналізатора, поліпшення мозкового кровообігу, подолання несприятливих наслідків гіподинамії, доцільно деякі перерви використовувати для виконання комплексу спеціальних профілактично-реабілітаційних вправ. За умови високого рівня напруженості робіт з ВДТ доцільним також слід вважати психофізіологічне розвантаження у спеціально обладнаних приміщеннях (кімнатах психофізіологічного розвантаження) під час регламентованих перерв або в кінці робочого дня. [81]

Окрім ДНАОП 0.00-1.31-99 та ДСанПіН 3.3.2-007-98, які регламентують вимоги безпеки та санітарно-гігієнічні вимоги до обладнання робочих місць користувачів ВДТ, є ще ціла низка нормативних актів загального призначення, які необхідно враховувати під час організації роботи користувачів ВДТ. Важливим нормативним актом є "Гігієнічна класифікація умов праці за показниками шкідливості та небезпечності факторів виробничого середовища, важності та напруженості трудового процесу", затверджена наказом Міністерства охорони здоров'я від 31. 12. 97 № 382. Гігієнічна класифікація праці необхідна для оцінки конкретних умов та характеру праці на робочих місцях. На основі такої оцінки приймаються рішення, спрямовані на запобігання або максимальне обмеження впливу несприятливих виробничих факторів. [82]

6.2 Електробезпека та пожежна безпека в приміщеннях з персональними комп'ютерами.

Вимоги електробезпеки у приміщеннях, де встановлені електронно-обчислювальні машини і персональні комп'ютери (далі — ЕОМ) відображені у ДНАОП 0.00-1.31-99. Відповідно до цього нормативного документу під час проектування систем електропостачання, монтажу основного електрообладнання та електричного освітлення будівель та приміщень для ЕОМ необхідно дотримуватись вимог Правил влаштування електроустановок (ПВЕ) ГОСТ 12.1.006-84, ГОСТ 12.1.019-79, ГОСТ 12.1.030-81, ГОСТ 12.1.045-84, ПТЕ, ПВЕ, ВСН 59-88 "Электрооборудование жилых и общественных зданий. Нормы проектирования", СН 357-77 "Инструкция по проектированию силового осветительного оборудования промышленных предприятий", Правил пожежної безпеки в Україні та інших нормативних документів, що стосуються штучного освітлення і електротехнічних пристроїв, а також вимог нормативно-технічної експлуатаційної документації заводу-виробника. ЕОМ, периферійні пристрої ЕОМ та устаткування для обслуговування, ремонту та налагодження ЕОМ, інше устаткування (апарати управління, контрольно-вимірювальні прилади, світильники тощо), електропроводи та кабелі за виконанням та ступенем захисту мають відповідати класу зони за ПВЕ, мати апаратуру захисту від струму короткого замикання та інших аварійних режимів.

Під час монтажу та експлуатації ліній електромережі необхідно повністю унеможливити виникнення електричного джерела загорання внаслідок короткого замикання та перевантаження проводів, обмежувати застосування проводів з легкозаймистою ізоляцією і, за можливості, перейти на негорючу ізоляцію.

Лінія електромережі для живлення ЕОМ, периферійних пристроїв ЕОМ та устаткування для обслуговування, ремонту та налагодження ЕОМ виконується як окрема групова трипровідна мережа, шляхом прокладання фазового, нульового робочого та нульового захисного провідників. Нульовий захисний провідник

використовується для заземлення (занулення) електроприймачів і прокладається від стійки групового розподільчого щита, розподільчого пункту до розеток живлення.

Використання нульового робочого провідника як нульового захисного провідника забороняється, а також не допускається підключення цих провідників на щиті до одного контактного затискача.

Площа перерізу нульового робочого та нульового захисного провідника в груповій трипровідній мережі повинна бути на менше площі перерізу фазового провідника. Усі провідники повинні відповідати номінальним параметрам мережі та навантаження, умовам навколишнього середовища, умовам розподілу провідників, температурному режиму та типам апаратури захисту, вимогам ПВЕ.

У приміщенні, де одночасно експлуатується або обслуговується більше п'яти персональних ЕОМ, на помітному та доступному місці встановлюється аварійний резервний вимикач, який може повністю вимкнути електричне живлення приміщення, крім освітлення.

ЕОМ, периферійні пристрої ЕОМ та устаткування для обслуговування, ремонту та налагодження ЕОМ повинні підключатися до електромережі тільки за допомогою справних штепсельних з'єднань і електророзеток заводського виготовлення. [82]

Штепсельні з'єднання та електророзетки крім контактів фазового та нульового робочого провідників повинні мати спеціальні контакти для підключення нульового захисного провідника. Конструкція їх має бути такою, щоб приєднання нульового захисного провідника відбувалося раніше ніж приєднання фазового та нульового робочого провідників. Порядок роз'єднання при відключенні має бути зворотним. Необхідно унеможливити з'єднання контактів фазових провідників з контактами нульового захисного провідника.

Неприпустимим є підключення ЕОМ, периферійних пристроїв ЕОМ та устаткування для обслуговування, ремонту та налагодження ЕОМ до звичайної

двопровідної електромережі, в тому числі — з використанням перехідних пристроїв.

Електромережі штепсельних з'єднань та електророзеток для живлення персональних ЕОМ, периферійних пристроїв ЕОМ та устаткування для обслуговування, ремонту та налагодження ЕОМ слід виконувати за магістральною схемою, по 3—6 з'єднань або електророзеток в одному колі.

Штепсельні з'єднання та електророзетки для напруги 12 В та 36 В за своєю конструкцією повинні відрізнятися від штепсельних з'єднань для напруги 127 В та 220 В. Окрім того вони мають бути пофарбовані в колір, який візуально значно відрізняється від кольору штепсельних з'єднань, розрахованих на напругу 127 В та 220 В.

Індивідуальні та групові штепсельні з'єднання та електророзетки необхідно монтувати на негорючих або важкогорючих пластинах з урахуванням вимог ПВЕ та Правил пожежної безпеки в Україні.

Електромережу штепсельних розеток для живлення персональних ЕОМ, периферійних пристроїв ЕОМ та устаткування для обслуговування, ремонту та налагодження ЕОМ при розташуванні їх уздовж стін приміщення прокладають по підлозі поряд зі стінами приміщення, як правило, в металевих трубах і гнучких металевих рукавах з відводами відповідно до затвердженого плану розміщення обладнання та технічних характеристик обладнання.

При розташуванні в приміщенні за його периметром до 5 персональних ЕОМ, використанні трипровідникового захищеного проводу або кабелю в оболонці з негорючого або важкогорючого матеріалу дозволяється прокладання їх без металевих труб та гнучких металевих рукавів.

Електромережу штепсельних розеток для живлення персональних ЕОМ, периферійних пристроїв ЕОМ та устаткування для обслуговування, ремонту та налагодження ЕОМ при розташуванні їх у центрі приміщення, прокладають у каналах або під знімною підлогою в металевих трубах або в гнучких металевих рукавах. При цьому не дозволяється застосовувати провід і кабель в ізоляції з

вулканізованої гуми та інші матеріали, що містять сірку. Відкрита прокладка кабелів під підлогою забороняється.

Металеві труби та гнучкі металеві рукави повинні бути заземлені. Заземлення повинно відповідати вимогам ДНАОП 0.00-1.21-98 "Правила безпечної експлуатації електроустановок споживачів". Заземлені конструкції, що знаходяться у приміщеннях (батареї опалення, водопровідні труби, кабелі із заземленим відкритим екраном тощо), мають бути надійно захищені діелектричними щитками або сітками від випадкового дотику.

Конструкція знімної підлоги повинна бути такою, щоб забезпечувались:

- вільний доступ до кабельних комунікацій під час обслуговування; |
- стійкість до горизонтальних зусиль при частково знятих плитах;
- вирівнювання поверхні підлоги за допомогою регулювальних опорних елементів;
- взаємозамінюваність плит.

Отвори в плитах для прокладання кабелів електроживлення виконуються, безпосередньо в місцях встановлення устаткування відповідно до затвердженого, технологічного плану розміщення устаткування та його технічних характеристик.

Для підключення переносної електроапаратури застосовують гнучкі проводи в надійній ізоляції. Тимчасова електропроводка від переносних приладів до джерел живлення виконується найкоротшим шляхом без заплутування проводів у конструкціях машин, приладів та меблях. Доточувати проводи можна тільки » шляхом паяння з наступним старанним ізолюванням місць з'єднання. [81]

Є неприпустимими:

- експлуатація кабелів та проводів з пошкодженою або такою, що втратила захисні властивості за час експлуатації, ізоляцією; залишення під напругою кабелів та проводів з неізолюваними провідниками;
- застосування саморобних продовжувачів, які не відповідають вимогам ПВЕ, до переносних електропроводок;
- застосування для опалення приміщення нестандартного (саморобного)

електронагрівального обладнання або ламп розжарювання;

- користування пошкодженими розетками, розгалужувальними та з'єднувальними коробками, вимикачами та іншими електровиробами, а також лампами, скло яких має сліди затемнення або випинання;

- підвішування світильників безпосередньо на струмопровідних проводах, обгортання електроламп і світильників папером, тканинами та іншими горючими матеріалами, експлуатація їх зі знятими ковпаками (розсіювачами);

- використання електроапаратури та приладів в умовах, що не відповідають вказівкам (рекомендаціям) підприємств-виготовлювачів.

На закінчення необхідно зазначити, що дотримання вищезазначених вимог значно підвищує електробезпеку, однак не може стовідсотково гарантувати неможливість ураження користувача електричним струмом. З огляду на це, необхідно знати і вміти правильно надавати першу допомогу при ураженні людини електричним струмом.

Залежно від особливостей виробничого процесу, крім загальних вимог пожежної безпеки, здійснюються спеціальні протипожежні заходи для окремих видів виробництв, технологічних процесів та промислових об'єктів. Для споруд та приміщень, в яких експлуатуються відеотермінали та ЕОМ, такі заходи визначені Правилами пожежної безпеки в Україні, ДНАОП 0.00-1.31-99 та іншими нормативними документами.

Будівлі і ті їх частини, в яких розташовуються ЕОМ, повинні бути не нижче II ступеня вогнестійкості. Над та під приміщеннями, де розташовуються ЕОМ, а також у суміжних з ними приміщеннях не дозволяється розташування приміщень категорій А і Б за вибухопожежною небезпекою. Приміщення категорії В слід відділяти від приміщень з ЕОМ протипожежними стінами.

Для всіх споруд і приміщень, в яких експлуатуються відеотермінали та ЕОМ, повинна бути визначена категорія з вибухопожежної і пожежної небезпеки відповідно до ОНТП 24-86 "Определения категорий помещений и зданий по взрывопожарной и пожарной опасности", та клас зони згідно з Правилами

влаштування електроустановок. Відповідні позначення повинні бути нанесені на вхідні двері приміщення.

Сховища інформації, приміщення для зберігання перфокарт, магнітних стрічок, пакетів магнітних дисків слід розміщати у відокремлених приміщеннях, обладнаних негорючими стелажими і шафами. Зберігати такі носії інформації на стелажах необхідно в металевих касетах. В приміщеннях ЕОМ слід зберігати лише ті носії інформації, які необхідні для поточної роботи.

Фальшпідлога у приміщеннях ЕОМ повинна бути виготовлена з негорючих матеріалів (або важкогорючих з межею вогнестійкості не менше 0,5 год.). Простір під знімною підлогою розділяють негорючими діафрагмами на відсіки площею не більше 250 м². Межа вогнестійкості діафрагми повинна бути не меншою за 0,75 год. Комунікації прокладають крізь діафрагми в спеціальних обоймах із застосуванням негорючих ущільнювачів для запобігання проникненню вогню з одного відсіку в інший, а також з міжпідлогового простору в приміщення. Міжпідлоговий простір під знімною підлогою має бути оснащений системою автоматичної пожежної сигналізації та засобами пожежогасіння відповідно до вимог Переліку однотипних за призначенням об'єктів, які підлягають обладнанню автоматичними установками пожежогасіння та пожежної сигналізації, СНиП 2.04.09-84, з використанням димових пожежних сповіщувачів.

Звукопоглинальне облицювання стін та стель у приміщеннях ЕОМ слід виготовляти з негорючих або важкогорючих матеріалів.

Для промивання деталей необхідно застосовувати негорючі миючі препарати. Промивання чарунок та інших знімних пристроїв горючими рідинами дозволяється лише у спеціальних приміщеннях, обладнаних припливно-витяжною вентиляцією. У випадку необхідності проведення дрібного ремонту або технічного обслуговування ЕОМ безпосередньо в машинному залі та при неможливості застосування негорючих миючих речовин дозволяється мати не більше 0,5 л легкозаймистої рідини у тарі, що не б'ється та щільно закривається.

Приміщення, в яких розташовуються персональні ЕОМ та дисплейні зали, повинні бути оснащені системою автоматичної пожежної сигналізації з димовими пожежними сповіщувачами та переносними вуглекислотними вогнегасниками з розрахунку 2 шт. на кожні 20 м² площі приміщення з урахуванням гранично допустимих концентрацій вогнегасної речовини. [81]

Не рідше одного разу на квартал необхідно очищати від пилу агрегати та вузли, кабельні канали та простір між підлогами.

6.3 Причини виникнення, загальна характеристика та класифікація надзвичайних ситуацій

Щодня в світі фіксуються тисячі подій, при яких відбувається порушення нормальних умов життя і діяльності людей, і які можуть призвести або призводять до загибелі людей та/або до значних матеріальних втрат. Такі події називаються надзвичайними ситуаціями.

Засоби масової інформації, як правило, привертають увагу громадськості до надзвичайних ситуацій, особливо, коли вони пов'язані з життям відомих особистостей, призвели або можуть призвести до великої кількості жертв, становлять загрозу нормальному життю і діяльності груп людей, цілих регіонів чи навіть країн. Майже жодне газетне видання, жоден випуск радіо або телевізійних новин не виходить без таких повідомлень.

Небезпека — це негативна властивість матерії, яка проявляється у здатності її завдавати шкоди певним елементам Всесвіту, потенційне джерело шкоди. Якщо мова йде про небезпеку для людини, то - це явища, процеси, об'єкти, властивості, здатні за певних умов завдавати шкоди здоров'ю чи життю людини або системам, що забезпечують життєдіяльність людей.

Загальні ознаки надзвичайних ситуацій (НС), види НС:

- наявність або загрози загибелі людей чи значне погіршення умов їх життєдіяльності;

- заподіяння економічних збитків;
- істотне погіршення стану довкілля.

До надзвичайних ситуацій, як правило, призводять аварії, катастрофи, стихійні лиха та інші події, такі як епідемії, терористичні акти, збройні конфлікти тощо.

Аварії поділяються на дві категорії:

- до I категорії належать аварії, внаслідок яких: загинуло 5 чи травмовано 10 і більше осіб; стався викид отруйних, радіоактивних, біологічно небезпечних речовин за санітарно-захисну зону підприємства; збільшилась концентрація забруднюючих речовин у навколишньому природному середовищі більш як у 10 разів; зруйновано будівлі, споруди чи основні конструкції об'єкта, що створило загрозу для життя і здоров'я значної кількості працівників підприємства чи населення;

- до II категорії належать аварії, внаслідок яких: загинуло до 5 чи травмовано від 4 до 10 осіб; зруйновано будівлі, споруди чи основні конструкції об'єкта, що створило загрозу для життя і здоров'я працівників цеху, діляниці (враховуються цех, діляниця з чисельністю працівників 100 осіб і більше).

Випадки порушення технологічних процесів, роботи устаткування, тимчасової зупинки виробництва в результаті спрацювання автоматичних захисних блокувань та інші локальні порушення у роботі цехів, діляниць і окремих об'єктів, падіння опор та обрив дротів ліній електропередач не належать до аварій, що мають категорії. [81]

Події природного походження або результат діяльності природних процесів, які за своєю інтенсивністю, масштабом поширення і тривалістю можуть вражати людей, об'єкти економіки та довкілля, називаються небезпечними природними явищами. Руйнівне небезпечне природне явище — це стихійне лихо.

Надзвичайні ситуації мають різні масштаби за кількістю жертв, кількістю людей, що стали хворими чи каліками, кількістю людей, яким завдано моральної

шкоди, за розмірами економічних збитків, площею території, на якій вони розвивались, тощо.

Вагомість надзвичайної ситуації визначається передусім кількістю жертв та ступенем впливу на оточуюче життєве середовище, тобто рівнем системи «людина — життєве середовище» (далі — «Л — ЖС»), якої вона торкнулася, і розміром шкоди, спричиненої цій системі. Виходячи з ієрархії систем «Л — ЖС», можна говорити про:

- індивідуальні надзвичайні ситуації, коли виникає загроза для порушення життєдіяльності лише однієї особи;
- надзвичайні ситуації рівня мікроколективу, тобто коли Загроза їх виникнення чи розповсюдження наслідків стосується сім'ї, виробничої бригади, пасажирів одного купе тощо;
- надзвичайні ситуації рівня колективу;
- надзвичайні ситуації рівня макроколективу;
- надзвичайні ситуації для жителів міста, району;
- надзвичайні ситуації для населення області;
- надзвичайні ситуації для населення країни;
- надзвичайні ситуації для жителів континенту;
- надзвичайні ситуації для всього людства.

Як правило, чим більшу кількість людей обходить надзвичайна ситуація, тим більшу територію вона охоплює. І навпаки, при більшій площі поширення катастрофи чи стихійного лиха від нього страждає більша кількість людей. Через це в основу існуючих класифікацій надзвичайних ситуацій за їх масштабом найчастіше кладуть територіальний принцип, за яким надзвичайні ситуації поділяють на локальні, об'єктові, місцеві, регіональні, загальнодержавні (національні), континентальні та глобальні (загальнопланетарні).

Локальні надзвичайні ситуації відповідають рівню системи «Л — ЖС» з однією особою та мікроколективом; об'єктові — системам з рівнем колектив,

макроколектив; місцеві — системам, в які входить населення міста або району; регіональні — області; загальнодержавні — населення країни і так далі.

Сьогоднішня ситуація в Україні щодо небезпечних природних явищ, аварій і катастроф характеризується як дуже складна. Тенденція зростання кількості надзвичайних ситуацій, важкість їх наслідків змушують розглядати їх як серйозну загрозу безпеці окремої людини, суспільству та навколишньому середовищу, а також стабільності розвитку економіки країни. До роботи в районі надзвичайної ситуації необхідно залучати значну кількість людських, матеріальних і технічних ресурсів. Запобігання надзвичайним ситуаціям, ліквідація їх наслідків, максимальне зниження масштабів втрат та збитків перетворилося на загальнодержавну проблему і є одним з найважливіших завдань органів виконавчої влади і управління всіх рівнів.

15 липня 1998 р. Постановою Кабінету Міністрів України № 1099 «Про порядок класифікації надзвичайних ситуацій» затверджено «Положення про класифікацію надзвичайних ситуацій».

Рівні надзвичайних ситуацій:

- загальнодержавний;
- регіональний;
- місцевий;
- об'єктовий.

В залежності від кількості людей, які загинули, розрізняють чотири рівні надзвичайних ситуацій.

Надзвичайна ситуація загальнодержавного рівня — це надзвичайна ситуація, яка розвивається на території двох та більше областей (Автономної Республіки Крим, міст Києва та Севастополя) або загрожує транскордонним перенесенням, а також у разі, коли для її ліквідації необхідні матеріали і технічні ресурси в обсягах, що перевищують власні можливості окремої області (Автономної Республіки Крим, міст Києва та Севастополя), але не менше одного відсотка обсягу видатків відповідного бюджету.

Надзвичайна ситуація регіонального рівня — це надзвичайна ситуація, яка розвивається на території двох або більше адміністративних районів (міст обласного значення) Автономної Республіки Крим, областей, міст Києва та Севастополя або загрожує перенесенням на територію суміжної області України, а також у разі, коли для її ліквідації необхідні матеріальні і технічні ресурси в обсягах, що перевищують власні можливості окремого району, але не менше одного відсотка обсягу видатків відповідного бюджету.

Надзвичайна ситуація місцевого рівня — це надзвичайна ситуація, яка виходить за межі потенційно-небезпечного об'єкта, загрожує поширенням самої ситуації або її вторинних наслідків на довкілля, сусідні населені пункти, інженерні споруди, а також у разі, коли для її ліквідації необхідні матеріальні і технічні ресурси в обсягах, що перевищують власні можливості потенційно-небезпечного об'єкта, але не менше одного відсотка обсягу видатків відповідного бюджету. До місцевого рівня також належать всі надзвичайні ситуації, які виникають на об'єктах житлово-комунальної сфери та інших, що не входять до затверджених переліків потенційно небезпечних об'єктів.

Надзвичайна ситуація об'єктового рівня — це надзвичайна ситуація, яка не підпадає під зазначені вище визначення, тобто така, що розгортається на території об'єкта або на самому об'єкті і наслідки якої не виходять за межі об'єкта або його санітарно-захисної зони.

Для організації ефективної роботи із запобігання надзвичайним ситуаціям, ліквідації їхніх наслідків, зниження масштабів втрат та збитків дуже важливо знати причини їх виникнення та володіти теорією виникнення катастроф.

Положення про класифікацію надзвичайних ситуацій за характером походження подій, котрі зумовлюють виникнення надзвичайних ситуацій на території України, розрізняє чотири класи надзвичайних ситуацій — надзвичайні ситуації техногенного, природного, соціально-політичного, військового характеру. Кожен клас надзвичайних ситуацій поділяється на групи, які містять конкретні їх види.

Надзвичайні ситуації техногенного характеру — це транспортні аварії (катастрофи), пожежі, неспровоковані вибухи чи їх загроза, аварії з викидом (загрозою викиду) небезпечних хімічних, радіоактивних, біологічних речовин, раптове руйнування споруд та будівель, аварії на інженерних мережах і спорудах життєзабезпечення, гідродинамічні аварії на греблях, дамбах тощо.

Надзвичайні ситуації природного характеру — це небезпечні геологічні, метеорологічні, гідрологічні морські та прісноводні явища, деградація ґрунтів чи надр, природні пожежі, зміна стану повітряного басейну, інфекційна захворюваність людей, сільськогосподарських тварин, масове ураження сільськогосподарських рослин хворобами чи шкідниками, зміна стану водних ресурсів та біосфери тощо.

Надзвичайні ситуації соціально-політичного характеру — це ситуації, пов'язані з протиправними діями терористичного та антиконституційного спрямування: здійснення або реальна загроза терористичного акту (збройний напад, захоплення і затримання важливих об'єктів, ядерних установок і матеріалів, систем зв'язку та телекомунікацій, напад чи замах на екіпаж повітряного чи морського судна), викрадення (спроба викрадення) чи знищення суден, встановлення вибухових пристроїв у громадських місцях, викрадення або захоплення зброї, виявлення застарілих боєприпасів тощо.

Надзвичайні ситуації воєнного характеру — це ситуації, пов'язані з наслідками застосування зброї масового ураження або звичайних засобів ураження, під час яких виникають вторинні фактори ураження населення внаслідок зруйнування атомних і гідроелектричних станцій, складів і сховищ радіоактивних і токсичних речовин та відходів, нафтопродуктів, вибухівки, сильнодіючих отруйних речовин, токсичних відходів, нафтопродуктів, вибухівки, транспортних та інженерних комунікацій тощо.

Сьогоднішня ситуація в Україні щодо небезпечних природних явищ, аварій і катастроф характеризується як дуже складна. Тенденція зростання кількості надзвичайних ситуацій, важкість їх наслідків змушують розглядати їх як серйозну

загрозу безпеці окремої людини, суспільству та навколишньому середовищу, а також стабільності розвитку економіки країни. [82]

До роботи в районі надзвичайної ситуації необхідно залучати значну кількість людських, матеріальних і технічних ресурсів. Запобігання надзвичайним ситуаціям, ліквідація їх наслідків, максимальне зниження масштабів втрат та збитків перетворилося на загальнодержавну проблему і є одним з найважливіших завдань органів виконавчої влади і управління всіх рівнів.

ВИСНОВКИ

В результаті виконання цієї дипломної роботи було створено сервіс на основі багатоварової клієнт-серверної архітектури, який спрощує та удосконалює обробку великих обсягів інформації, зокрема, за рахунок зручної маніпуляції статистичними даними.

В результаті аналізу предметної області було встановлено, що багатоварова клієнт-серверна архітектура підходить для цього сервісу найкраще, а також було спроектовано основний алгоритм програми

Було встановлено мету створення програми, визначено основні функції. Встановлено вимоги до надійності та спроектованої системи, а також визначені умови роботи та розповсюдження програми

Було обґрунтовано середовища розробки та функціонування програми, Створено раніше спроектований основний алгоритм. Описані всі модулі програми з описом технологій які були використані з доказом вибору головних із них, а саме Angular і NestJS за допомогою порівняння їх з іншими.

Було створено методику роботи користувача з системою окремо для інших розробників, окремо для звичайних користувачів

Проведено розрахунок основної та додаткової заробітної плати фахівців, зайнятих у розробці програмного продукту. Загальні витрати, пов'язані з розробкою програмного продукту, складають 105 019,73 грн.

В процесі виконання дипломної роботи були опрацьовані правові аспекти охорони праці користувачів комп'ютерів, розглянуті питання електробезпеки, пожежної безпеки в приміщеннях з персональними комп'ютерами, дана загальна характеристика надзвичайних ситуацій.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1 Інформаційний вибух і глобалізація світової політики [Електронний ресурс]. – Режим доступу:

https://enigma.ua/articles/informatsiyniy_vibukh_i_globalizatsiya_svitovoi_politiki

2 Основи теорії систем і системного аналізу [Електронний ресурс]. – Режим доступу:

https://eprints.kname.edu.ua/10895/1/%D0%A1%D0%B8%D1%81%D0%90%D0%BD%D0%B0%D0%BB%D0%B8%D0%B7_1_8%D0%BD.pdf

3 Порівняльний аналіз сучасних методологій розробки програмного забезпечення [Електронний ресурс]. – Режим доступу:

http://www.rusnauka.com/40_OINBG_2014/Informatica/3_182487.doc.htm

4 Автентифікація в кібербезпеці [Електронний ресурс]. – Режим доступу:

https://er.nau.edu.ua/bitstream/NAU/46983/1/%D0%A4%D0%9A%D0%9A%D0%9F%D0%86_2020_125_%D0%9A%D0%BE%D0%BB%D1%8F%D0%BA%D0%B0%D0%90%D0%92.pdf

5. Server Architectures: Multiprocessors, Clusters, Parallel Systems, Web Servers, Storage Solutions / Ren Chevance – Elsevier/Digital Press, 2005. – 784 с.

6 Desktop Browser Market Share Worldwide [Електронний ресурс]. – Режим доступу: <https://gs.statcounter.com/browser-market-share/desktop/worldwide>

7 Most used web frameworks among developers worldwide, as of 2022 [Електронний ресурс]. – Режим доступу:

<https://www.statista.com/statistics/1124699/worldwide-developer-survey-most-used-frameworks-web/>

8 Visual Studio Code [Електронний ресурс]. – Режим доступу: <https://code.visualstudio.com/docs>

9 Git Pro. Getting started about version control [Електронний ресурс]. – Режим доступу: <https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control>

- 10 Git Pro. History of Git. [Електронний ресурс]. – Режим доступу: <https://git-scm.com/book/en/v2/Getting-Started-A-Short-History-of-Git>
- 11 Графік популярності систем контролю версій [Електронний ресурс]. – Режим доступу: <https://survey.stackoverflow.co/2022/#section-version-control-version-control-systems>
- 12 Використання Postman в тестуванні [Електронний ресурс]. – Режим доступу: <https://training.qatestlab.com/blog/technical-articles/use-postman-in-testing/>
- 13 Офіційний веб-сайт Postman [Електронний ресурс]. – Режим доступу: <https://www.postman.com/>
- 14 Документація Swagger UI [Електронний ресурс]. – Режим доступу: <https://swagger.io/tools/swagger-ui/>
- 15 Що таке браузер від Firefox [Електронний ресурс]. – Режим доступу: <https://www.mozilla.org/uk/firefox/browsers/what-is-a-browser/>
- 16 PWA [Електронний ресурс]. – Режим доступу: https://kneu.edu.ua/userfiles/zb_mise/99/8.pdf
- 17 Практичний приклад PWA [Електронний ресурс]. – Режим доступу: <https://medium.com/dev-channel/a-pinterest-progressive-web-app-performance-case-study-3bd6ed2e6154>
- 18 І. О. Бардус, М. І. Лазарєв, А. О. Ніценко БАЗИ ДАНИХ У СХЕМАХ (НА ОСНОВІ ФУНДАМЕНТАЛІЗОВАНОГО ПІДХОДУ) - Діса Плюс, 2017 - 133с.
- 19 MongoDB [Електронний ресурс]. – Режим доступу: <https://www.geeksforgeeks.org/what-is-mongodb-working-and-features/>
- 20 JavaScript [Електронний ресурс]. – Режим доступу: <https://developer.mozilla.org/ru/docs/conflicting/Web/JavaScript>
- 21 TypeScript [Електронний ресурс]. – Режим доступу: <https://www.typescriptlang.org/>
- 22 The state of open source software [Електронний ресурс]. – Режим доступу: <https://octoverse.github.com>

- 23 NodeJS [Електронний ресурс]. – Режим доступу: <https://nodejs.org/en/about/>
- 24 NVM [Електронний ресурс]. – Режим доступу: <https://github.com/nvm-sh/nvm>
- 25 NPM [Електронний ресурс]. – Режим доступу: <https://www.npmjs.com/about>
- 26 PNPM [Електронний ресурс]. – Режим доступу: <https://pnpm.io/motivation>
- 27 HTML [Електронний ресурс]. – Режим доступу: <https://developer.mozilla.org/en-US/docs/Web/HTML>
- 28 Frontend vs Backend [Електронний ресурс]. – Режим доступу: <https://dan-it.com.ua/uk/blog/rozrobka-z-boku-front-end-shho-ce-take-i-chim-vidriznjaietsja-vid-back-end/>
- 29 Framework [Електронний ресурс]. – Режим доступу: <https://tonyline.com.ua/glossary/framework/>
- 30 Angular [Електронний ресурс]. – Режим доступу: <https://angular.io/docs>
- 31 SPA офіційна документація [Електронний ресурс]. – Режим доступу: <https://developer.mozilla.org/en-US/docs/Glossary/SPA>
- 32 Що таке SPA більш детально [Електронний ресурс]. – Режим доступу: <https://wezom.com.ua/ua/blog/chto-takoe-spa-prilozheniya>
- 33 React [Електронний ресурс]. – Режим доступу: <https://legacy.reactjs.org/>
- 34 Vue [Електронний ресурс]. – Режим доступу: <https://ua.vuejs.org/guide/introduction.html#what-is-vue>
- 35 React JSX [Електронний ресурс]. – Режим доступу: <https://react.dev/learn/writing-markup-with-jsx>
- 36 Data Binding [Електронний ресурс]. – Режим доступу: <https://www.techtarget.com/whatis/definition/data-binding>
- 37 Angular types of data binding [Електронний ресурс]. – Режим доступу: <https://angular.io/guide/binding-syntax#types-of-data-binding>

- 38 Decorator [Электронный ресурс]. – Режим доступа:
<https://refactoring.guru/uk/design-patterns/decorator>
- 39 Dependency injection [Электронный ресурс]. – Режим доступа:
<https://angular.io/guide/architecture-services#dependency-injection-di>
- 40 Angular Architecture modules [Электронный ресурс]. – Режим доступа:
<https://angular.io/guide/architecture-modules>
- 41 Angular component overview [Электронный ресурс]. – Режим доступа:
<https://angular.io/guide/component-overview>
- 42 Angular Pipes [Электронный ресурс]. – Режим доступа:
<https://angular.io/guide/glossary#pipe>
- 43 Angular Attribute directives [Электронный ресурс]. – Режим доступа:
<https://angular.io/guide/attribute-directives>
- 44 TailwindCss [Электронный ресурс]. – Режим доступа:
<https://tailwindcss.com/>
- 45 RxJs [Электронный ресурс]. – Режим доступа:
<http://reactivex.io/rxjs/manual/overview.html>
- 46 NGRX Store [Электронный ресурс]. – Режим доступа:
<https://ngrx.io/guide/store/why>
- 47 NGRX Effects [Электронный ресурс]. – Режим доступа:
<https://ngrx.io/guide/effects>
- 48 Angular svg icon [Электронный ресурс]. – Режим доступа:
<https://www.npmjs.com/package/angular-svg-icon>
- 49 Lottie [Электронный ресурс]. – Режим доступа: <https://airbnb.io/lottie/>
- 50 Ngx-lottie [Электронный ресурс]. – Режим доступа:
<https://www.npmjs.com/package/ngx-lottie>
- 51 Angular material CDK [Электронный ресурс]. – Режим доступа:
<https://material.angular.io/cdk/categories>
- 52 Angular material cdk drag and drop [Электронный ресурс]. – Режим доступа:
<https://material.angular.io/cdk/drag-drop/overview>

- 53 Chart.js [Электронный ресурс]. – Режим доступа:
<https://www.chartjs.org/docs/latest/>
- 54 Google Identity [Электронный ресурс]. – Режим доступа:
<https://cloud.google.com/identity-platform>
- 55 NestJS [Электронный ресурс]. – Режим доступа: <https://docs.nestjs.com/>
- 56 ExpressJS [Электронный ресурс]. – Режим доступа: <https://expressjs.com/>
- 57 Middlewares [Электронный ресурс]. – Режим доступа:
<https://expressjs.com/en/guide/using-middleware.html>
- 58 Middlewares in express [Электронный ресурс]. – Режим доступа:
<https://iq.opengenus.org/middlewares-in-express/>
- 59 NestJS lifecycle [Электронный ресурс]. – Режим доступа:
<https://slides.com/yariv-gilad/nest-js-request-lifecycle/fullscreen>
- 60 NestJS request lifecycle [Электронный ресурс]. – Режим доступа:
<https://docs.nestjs.com/faq/request-lifecycle#middleware>
- 61 MongoDB [Электронный ресурс]. – Режим доступа:
<https://aws.amazon.com/ru/documentdb/what-is-mongodb>
- 62 Mongoose [Электронный ресурс]. – Режим доступа:
<https://www.freecodecamp.org/news/introduction-to-mongoose-for-mongodb-d2a7aa593c57>
- 63 body-parser [Электронный ресурс]. – Режим доступа:
<https://www.npmjs.com/package/body-parser>
- 64 CORS [Электронный ресурс]. – Режим доступа:
<https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS>
- 65 Cors npm package [Электронный ресурс]. – Режим доступа:
<https://www.npmjs.com/package/cors>
- 66 Sessions [Электронный ресурс]. – Режим доступа:
<https://docs.nestjs.com/techniques/session>
- 67 Connect-mongo [Электронный ресурс]. – Режим доступа:
<https://www.npmjs.com/package/connect-mongo>

- 68 JWT [Електронний ресурс]. – Режим доступу: <https://jwt.io/introduction>
- 69 Passport [Електронний ресурс]. – Режим доступу: <https://docs.nestjs.com/recipes/passport>
- 70 NestJS validations [Електронний ресурс]. – Режим доступу: <https://docs.nestjs.com/techniques/validation>
- 71 class-validator [Електронний ресурс]. – Режим доступу: <https://www.npmjs.com/package/class-validator>
- 72 class-transformer [Електронний ресурс]. – Режим доступу: <https://www.npmjs.com/package/class-transformer>
- 73 Prettier [Електронний ресурс]. – Режим доступу: <https://prettier.io/docs/en/index.html>
- 74 ESLint [Електронний ресурс]. – Режим доступу: <https://dev.to/shivambmgupta/eslint-what-why-when-how-5f1d>
- 75 Л.Є. Довгань, Г.А. Мохонько, І.П. Малик. Управління проєктами – К.: КПІ ім. Ігоря Сікорського, 2017. – 420 с.
- 76 Герасимчук В.Г., Розенплентер А.Е. // Економіка та організація виробництва Підручник.- Київ: Знання, 2007. - 678 с.
- 77 Економіка підприємства: Підручник / За заг. ред. С.Ф. Покропивного. - Вид. 3-тє, перероб. та доп. - К.: КНЕУ, 2010. - 528 с.
- 78 Про оплату праці: Закон України від 24.03.95р. №108/95-ВР (зі змінами та доповненнями)
- 79 Про збір та облік єдиного внеску на загальнообов'язкове державне соціальне страхування: Закон України від 02.12.2010 № 2755-VI (зі змінами та доповненнями)
- 80 Економічне обґрунтування інноваційних рішень: навчальний посібник / В.В. Кавецький, І.В. Причепа, Л.О. Нікіфорова – Вінниця : ВНТУ, 2016. – 136 с.
- 81 Жидецький В.Ц. Охорона праці користувачів комп'ютерів. –Львів: Афіша, 2000. – 176 с.

82 Желібо Є.П., Заверуха Н.М., Зацарний В.В. Безпека життєдіяльності. – К.: Каравела, 2005. – 344 с.

ДОДАТОК А
(обов'язковий)
Текст програми

Frontend:

src\main.ts

```
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
import { AppModule } from './app/app.module';
platformBrowserDynamic().bootstrapModule(AppModule).catch(err => console.error(err))
```

src\app\app-routing.module.ts

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { HomeComponent } from './modules/home/home-page/home-page.component';
import { CategoriesPageComponent } from './modules/categories/categories-page/categories-page.component';
import { StatisticComponent } from './modules/statistic/statistic-page/statistic-page.component';
import { AuthorizationComponent } from './pages/authorization/authorization.component';
import { AuthGuardService } from './services/auth-guard.service';
import { PerfComponent } from './pages/perf/perf.component';
import { CategoryResolver } from './resolvers/category.resolver';
import { ColorResolver } from './resolvers/color.resolver';
import { StatisticResolver } from './resolvers/statistic.resolver';
const routes: Routes = [
  { path: '', component: HomeComponent, canActivate: [AuthGuardService], resolve: [ColorResolver, CategoryResolver] },
  { path: 'categories', component: CategoriesPageComponent, canActivate: [AuthGuardService], resolve: [ColorResolver, CategoryResolver] },
  { path: 'statistic', component: StatisticComponent, canActivate: [AuthGuardService], resolve: [ColorResolver, CategoryResolver, StatisticResolver] },
  { path: 'authorization', canActivate: [AuthGuardService], component: AuthorizationComponent },
  { path: 'perf', component: PerfComponent },
  { path: '', redirectTo: '/', pathMatch: 'full' }
]
@NgModule({ imports: [RouterModule.forRoot(routes, { useHash: true })], exports: [RouterModule] });
export class AppRoutingModuleModule { }
```

src\app\app.component.ts

```
import { Component, OnInit } from '@angular/core';
import { environment } from 'src/environments/environment';
import { User } from 'src/types/User';
import { AuthGuardService } from './services/auth-guard.service';
@Component({ selector: 'app-root', templateUrl: './app.component.html', styleUrls: ['./app.component.scss'] })
export class AppComponent implements OnInit {
  userData: User | null = null;
  initialize: boolean = false;
  title = 'counter-frontend';
  ngOnInit() {
    console.info(`Current version: ${environment.version}`);
    this.authGuard.authGuardData.subscribe(newUserData => {
      this.initialize = true;
      if (newUserData.authorized) {
        this.userData = newUserData;
        return;
      }
      this.userData = null;
    });
    constructor(private authGuard: AuthGuardService) {}
  }
}
```

src\app\app.module.ts

```
import { NgModule, isDevMode } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { HttpClientModule } from '@angular/common/http';
import { ReactiveFormsModule }
```

```

msModule } from '@angular/forms';import { SocialLoginModule,SocialAuthServiceConfig,GoogleLoginProvider } from '@abacritt/angularx-social-login';import { AppRoutingModuleModule } from './app-routing.module';import { AppComponent } from './app.component';import { AuthorizationComponent } from './pages/authorization/authorization.component';import { HeaderComponent } from './components/header/header.component';import { environment } from 'src/environments/environment';import { AngularSvgIconModule } from 'angular-svg-icon';import { LoaderComponent } from './components/loader/loader.component';import { StoreModule } from '@ngrx/store';import { PerfComponent } from './pages/perf/perf.component';import { StoreDevtoolsModule } from '@ngrx/store-devtools';import { EffectsModule } from '@ngrx/effects';import { StoreEffects,StoreReducers } from './store/';import { MenuComponent } from './components/menu/menu.component';import { LottieModule } from 'ngx-lottie';import { LogoComponent } from './components/logo/logo.component';import { LogoItemComponent } from './components/logo/logo-item/logo-item.component';import { FooterComponent } from './components/footer/footer.component';import { UserModule } from './modules/user/user.module';import { ClickedOutsideDirectiveModule } from './directives/clicked-outside-directive.module';import { CategoriesModule } from './modules/categories/categories.module';import { StatisticModule } from './modules/statistic/statistic.module';import { HomeModule } from './modules/home/home.module';import { ButtonsModule } from './modules/buttons/buttons.module';export function playerFactory(){return import('lottie-web')} @NgModule({ declarations:[AppComponent,AuthorizationComponent,HeaderComponent,LoaderComponent,PerfComponent,MenuComponent,LogoComponent,LogoItemComponent,FooterComponent], imports:[BrowserModule,AppRoutingModule,HttpClientModule,SocialLoginModule,ReactiveFormsModule,AngularSvgIconModule.forRoot(),StoreModule.forRoot(StoreReducers),EffectsModule.forRoot(StoreEffects),StoreDevtoolsModule.instrument({ maxAge:25,logOnly:!isDevMode() }),LottieModule.forRoot({ player:playerFactory }),UserModule,ClickedOutsideDirectiveModule,CategoriesModule,StatisticModule,HomeModule,ButtonsModule], providers:[ { provide:'SocialAuthServiceConfig',useValue: { autoLogin:false,providers:[ { id:GoogleLoginProvider.PROVIDER_ID,provider:new GoogleLoginProvider(environment.googleClientId,{ oneTapEnabled:false }) } ] },onError:err=>{ console.error(err) } } as SocialAuthServiceConfig ],bootstrap:[AppComponent] });export class AppModule {

```

```

src\app\components\footer\footer.component.ts
import { Component } from '@angular/core';import { environment } from 'src/environments/environment';import { GITHUB_CREATOR_ACCOUNT_LINK,USED_SOURCE_LINKS } from './footer.config'@Component({ selector:'app-footer',templateUrl:'./footer.component.html' });export class FooterComponent { version=environment.version;type=environment.type;config={ creatorLink:GITHUB_CREATOR_ACCOUNT_LINK,sourceLinks:USED_SOURCE_LINKS } }

```



```
src/app/components/footer/footer.config.ts
export const GITHUB_CREATOR_ACCOUNT_LINK='https:;;export const
USED_SOURCE_LINKS = [; {; icon: 'assets/icons/used-
social/tailwindcss.svg'; link: 'https:},{ icon:'assets/icons/used-
social/icons8.svg',link:']
```

```
src/app/components/header/header.component.ts
import { Component, Input } from '@angular/core'; import { User } from 'src/types/User' @Co
mponent({ selector: 'app-header', templateUrl: './header.component.html' }); export class
HeaderComponent { @Input() userInfo: User | null = null }
```

```
src/app/components/loader/loader.component.ts
import { Component } from '@angular/core' @Component({ selector: 'app-
loader', templateUrl: './loader.component.html', styleUrls: ['./loader.component.scss'] }); exp
ort class LoaderComponent { }
```

```
src/app/components/logo/logo.component.ts
import { Component } from '@angular/core'; import { LogoItemColors } from './logo-
item/logo-
item.types'; import { LOGO_ITEMS } from './logo.config'; import { LogoItem } from './logo.ty
pes' @Component({ selector: 'app-logo', templateUrl: './logo.component.html' }); export
class
LogoComponent { logoItems: LogoItem[]; constructor() { this.logoItems = LOGO_ITEMS }
}
```

```
src/app/components/logo/logo.config.ts
import { LogoItem } from './logo.types'; export const
LOGO_ITEMS: LogoItem[] = [ { delay: 0, color: 'rose', letter: 'C' }, { delay: 0.3, color: 'pink', lette
r: 'o' }, { delay: 0.6, color: 'fuchsia', letter: 'u' }, { delay: 0.9, color: 'purple', letter: 'n' }, { delay: 1.2, c
olor: 'violet', letter: 't' }, { delay: 1.5, color: 'indigo', letter: 'e' }, { delay: 1.8, color: 'blue', letter: 'r' } ]
```

```
src/app/components/logo/logo.types.ts
import { LogoItemColors } from './logo-item/logo-item.types'; export interface
LogoItem { delay: number; color: LogoItemColors; letter: string }
```

```
src/app/components/logo/logo-item/logo-item.component.ts
import { Component, ElementRef, Input, OnInit } from '@angular/core'; import { colors } from '
./logo-item.config'; import { LogoItemColors } from './logo-
item.types' @Component({ selector: 'app-logo-item', templateUrl: './logo-
item.component.html', styleUrls: ['./logo-item.component.scss'] }); export class
LogoItemComponent implements
OnInit { @Input() color: LogoItemColors = 'slate' @Input() delay: number = 0; backgroundGra
dient: string = ''; constructor(private
```

```

elRef:ElementRef){};ngOnInit():void{ console.log(this.color);const
GRADIENT_TYPE='linear-gradient';const GRADIENT_DIRECTION='to
bottom';const
GRADIENT_COLORS=[colors[this.color]['300'],colors[this.color]['400'],colors[this.col
or]['500'],colors[this.color]['600']].join(',');this.backgroundGradient=`background-
image:${GRADIENT_TYPE}(${GRADIENT_DIRECTION},${GRADIENT_COLOR
S})`;console.log(this.backgroundGradient);this.setAnimationDelayProperty(this.delay)}
;private setAnimationDelayProperty(delay:number){const delayNear=`${delay}s`const
delayDistant=`${delay-4}s`this.elRef.nativeElement.style.setProperty('--animation-
delay-near',delayNear);this.elRef.nativeElement.style.setProperty('--animation-delay-
distant',delayDistant)}}

```

```

src\app\components\logo\logo-item\logo-item.config.ts
import{LogoItemColorsConfig}from'./logo-item.types';export const
colors:LogoItemColorsConfig={slate:{'50':'#f8fafc','100':'#f1f5f9','200':'#e2e8f0','300':'#
cbd5e1','400':'#94a3b8','500':'#64748b','600':'#475569','700':'#334155','800':'#1e293b','9
00':'#0f172a'},gray:{'50':'#f9fafb','100':'#f3f4f6','200':'#e5e7eb','300':'#d1d5db','400':'#9c
a3af','500':'#6b7280','600':'#4b5563','700':'#374151','800':'#1f2937','900':'#111827'},zinc
:{'50':'#fafafa','100':'#f4f4f5','200':'#e4e4e7','300':'#d4d4d8','400':'#a1a1aa','500':'#71717
a','600':'#52525b','700':'#3f3f46','800':'#27272a','900':'#18181b'},neutral:{'50':'#fafafa','1
00':'#f5f5f5','200':'#e5e5e5','300':'#d4d4d4','400':'#a3a3a3','500':'#737373','600':'#52525
2','700':'#404040','800':'#262626','900':'#171717'},stone:{'50':'#fafaf9','100':'#f5f5f4','20
0':'#e7e5e4','300':'#d6d3d1','400':'#a8a29e','500':'#78716c','600':'#57534e','700':'#44403c
','800':'#292524','900':'#1c1917'},red:{'50':'#fef2f2','100':'#fee2e2','200':'#fecaca','300':'#
fca5a5','400':'#f87171','500':'#ef4444','600':'#dc2626','700':'#b91c1c','800':'#991b1b','90
0':'#7f1d1d'},orange:{'50':'#fff7ed','100':'#ffedd5','200':'#fed7aa','300':'#fdba74','400':'#f
b923c','500':'#f97316','600':'#ea580c','700':'#c2410c','800':'#9a3412','900':'#7c2d12'},am
ber:{'50':'#fffbeb','100':'#fef3c7','200':'#fde68a','300':'#fcd34d','400':'#fbbf24','500':'#f59
e0b','600':'#d97706','700':'#b45309','800':'#92400e','900':'#78350f'},yellow:{'50':'#fefce8
','100':'#fef9c3','200':'#fef08a','300':'#fde047','400':'#facc15','500':'#eab308','600':'#ca8a0
4','700':'#a16207','800':'#854d0e','900':'#713f12'},lime:{'50':'#f7fee7','100':'#ecfccb','200
':'#d9f99d','300':'#bef264','400':'#a3e635','500':'#84cc16','600':'#65a30d','700':'#4d7c0f','8
00':'#3f6212','900':'#365314'},green:{'50':'#f0fdf4','100':'#dcfce7','200':'#bbf7d0','300':'#
86efac','400':'#4ade80','500':'#22c55e','600':'#16a34a','700':'#15803d','800':'#166534','90
0':'#14532d'},emerald:{'50':'#ecfdf5','100':'#d1fae5','200':'#a7f3d0','300':'#6ee7b7','400':'
#34d399','500':'#10b981','600':'#059669','700':'#047857','800':'#065f46','900':'#064e3b'},
teal:{'50':'#f0fdfa','100':'#ccfbf1','200':'#99f6e4','300':'#5eead4','400':'#2dd4bf','500':'#14
b8a6','600':'#0d9488','700':'#0f766e','800':'#115e59','900':'#134e4a'},cyan:{'50':'#ecfeff','
100':'#cffffe','200':'#a5f3fc','300':'#67e8f9','400':'#22d3ee','500':'#06b6d4','600':'#0891b2
','700':'#0e7490','800':'#155e75','900':'#164e63'},sky:{'50':'#f0f9ff','100':'#e0f2fe','200':'#
bae6fd','300':'#7dd3fc','400':'#38bdf8','500':'#0ea5e9','600':'#0284c7','700':'#0369a1','800
':'#075985','900':'#0c4a6e'},blue:{'50':'#eff6ff','100':'#dbeafe','200':'#bfdbfe','300':'#93c5
fd','400':'#60a5fa','500':'#3b82f6','600':'#2563eb','700':'#1d4ed8','800':'#1e40af','900':'#1e
3a8a'},indigo:{'50':'#eef2ff','100':'#e0e7ff','200':'#c7d2fe','300':'#a5b4fc','400':'#818cf8','

```

```
500': '#6366f1', '600': '#4f46e5', '700': '#4338ca', '800': '#3730a3', '900': '#312e81'}, violet: { '50': '#f5f3ff', '100': '#ede9fe', '200': '#ddd6fe', '300': '#c4b5fd', '400': '#a78bfa', '500': '#8b5cf6', '600': '#7c3aed', '700': '#6d28d9', '800': '#5b21b6', '900': '#4c1d95'}, purple: { '50': '#faf5ff', '100': '#f3e8ff', '200': '#e9d5ff', '300': '#d8b4fe', '400': '#c084fc', '500': '#a855f7', '600': '#9333ea', '700': '#7e22ce', '800': '#6b21a8', '900': '#581c87'}, fuchsia: { '50': '#fdf4ff', '100': '#fae8ff', '200': '#f5d0fe', '300': '#f0abfc', '400': '#e879f9', '500': '#d946ef', '600': '#c026d3', '700': '#a21caf', '800': '#86198f', '900': '#701a75'}, pink: { '50': '#fdf2f8', '100': '#fce7f3', '200': '#fbcfe8', '300': '#f9a8d4', '400': '#f472b6', '500': '#ec4899', '600': '#db2777', '700': '#be185d', '800': '#9d174d', '900': '#831843'}, rose: { '50': '#fff1f2', '100': '#ffe4e6', '200': '#fecdd3', '300': '#fda4af', '400': '#fb7185', '500': '#f43f5e', '600': '#e11d48', '700': '#be123c', '800': '#9f1239', '900': '#881337' }
```

```
src/app/components/logo/logo-item/logo-item.types.ts
```

```
export type
```

```
LogoItemColors=|'slate'|'gray'|'zinc'|'neutral'|'stone'|'red'|'orange'|'amber'|'yellow'|'lime'|'green'|'emerald'|'teal'|'cyan'|'sky'|'blue'|'indigo'|'violet'|'purple'|'fuchsia'|'pink'|'rose';type
```

```
LogoItemColorInfo={ '50':string'100':string'200':string'300':string'400':string'500':string'600':string'700':string'800':string'900':string };export type
```

```
LogoItemColorsConfig=Record<LogoItemColors,LogoItemColorInfo>
```

```
src/app/components/menu/menu.component.ts
```

```
import { Component } from '@angular/core';import { AnimationItem } from 'lottie-
```

```
web';import { AnimationOptions } from 'ngx-
```

```
lottie';import { MENU_ITEMS } from './menu.config';import { AnimationOptionsWithId, AnimationSetStatusTypes, MenuItem } from './menu.types';import { getAnimationOptions } from './menu.helper'@Component({ selector: 'app-
```

```
menu', templateUrl: './menu.component.html' });export class
```

```
MenuComponent { menuItems: MenuItem[];private
```

```
animateOptions: AnimationOptionsWithId[];private
```

```
animationItems: AnimationItem[];constructor() { this.menuItems=MENU_ITEMS;this.animateOptions=getAnimationOptions(this.menuItems);this.animationItems=[] };getAnimateOptions(id:number) { const
```

```
animateOptions=this.animateOptions.find(item=>item.id===id);if(animateOptions===undefined) { console.error(`Not found animateOptions for id=${id}`);return { } };return
```

```
animateOptions };onAnimate(animationItem: AnimationItem):void { this.animationItems.push(animationItem) };private getAnimationItemById(id:number) { return
```

```
this.animationItems.find(item=>item.name===String(id)) };private
```

```
animationSetStatus(id:number,type: AnimationSetStatusTypes) { const
```

```
animationItem=this.getAnimationItemById(id);if(animationItem===undefined) { console.log(`No find animation with id=${id}`);return };switch(type) { case
```

```
AnimationSetStatusTypes.PLAY:animationItem.play();return;case
```

```
AnimationSetStatusTypes.STOP:animationItem.stop();return;default:console.log(`Not found animationSetStatus
```

```
type=${type}`);return } };mouseLinkEnter(id:number) { this.animationSetStatus(id,Anim
```

```

ationSetStatusTypes.PLAY));mouseLinkLeave(id:number){this.animationSetStatus(id,
AnimationSetStatusTypes.STOP)}}

src\app\components\menu\menu.config.ts
import { MenuItem } from './menu.types'; export const ICONS_PATH='assets/animated-
icons'; export const
MENU_ITEMS:MenuItem[]=[{id:0,name:'Home',path:'/',icon:'home'},{id:1,name:'Cate
gories',path:'/categories',icon:'opened-
folder'},{id:2,name:'Statistic',path:'/statistic',icon:'combo-chart'}]

src\app\components\menu\menu.helper.ts
import { AnimationOptions } from 'ngx-
lottie'; import { ICONS_PATH } from './menu.config'; import { AnimationOptionsWithId, Me
nuItem } from './menu.types'; const
defaultAnimateOptions:AnimationOptions={autoplay:false}; export const
getAnimationOptions=(menuItems:MenuItem[]):AnimationOptionsWithId[]=>{return
menuItems.map(item=>({name:String(item.id),...defaultAnimateOptions,id:item.id,path
: `${ICONS_PATH}/${item.icon}.json`)))}

src\app\components\menu\menu.types.ts
import { AnimationOptions } from 'ngx-lottie'; export interface
MenuItem { id:number; name:string; path:string; icon:string; } export type
AnimationOptionsWithId=AnimationOptions&{id:number}; export enum
AnimationSetStatusTypes {PLAY='play',STOP='stop'}

src\app\components-modules\load-status-button\load-status-button.component.ts
import { Component, EventEmitter, Input, Output } from '@angular/core'; import { LoadStatus
} from 'src/app/store/store.types'; import { ICONS } from './load-status-
button.config' @Component({ selector:'counter-load-status-button', templateUrl:'./load-
status-button.component.html', styleUrls:['./load-status-button.component.scss']}); export
class LoadStatusButtonComponent { @Output() buttonClick=new
EventEmitter() @Input() status:LoadStatus|null=LoadStatus.ERROR; icons=ICONS; onCl
ick(){this.buttonClick.emit()}}

src\app\components-modules\load-status-button\load-status-button.config.ts
import { LoadStatus } from 'src/app/store/store.types'; export const
ICONS=[ [LoadStatus.NOT_SYNCHRONIZED]:'assets/icons-
png/help.png', [LoadStatus.SYNCHRONIZATION]:'assets/icons-
png/upgrade.png', [LoadStatus.ERROR]:'assets/icons-
png/cancel.png', [LoadStatus.SYNCHRONIZED]:'assets/icons-png/ok.png']

src\app\components-modules\load-status-button\load-status-button.module.ts
import { NgModule } from '@angular/core'; import { CommonModule } from '@angular/com
mon'; import { LoadStatusButtonComponent } from './load-status-

```

```
button.component';import { ButtonsModule } from 'src/app/modules/buttons/buttons.module'@NgModule({ declarations:[LoadStatusButtonComponent],imports:[CommonModule,ButtonsModule],exports:[LoadStatusButtonComponent]});export class LoadStatusButtonModule{ }
```

```
src\app\components-modules\not-sync-status-icon\not-sync-status-icon.component.ts
import { Component, Input } from '@angular/core';import { NotSyncStatus } from 'src/app/store/types'@Component({ selector:'counter-not-sync-status-icon',templateUrl: './not-sync-status-icon.component.html' });export class NotSyncStatusIconComponent { @Input()status:NotSyncStatus|undefined=undefined;get statusIsNotSynchronized(){return this.status===NotSyncStatus.NOT_SYNCHRONIZED};get statusIsSynchronization(){return this.status===NotSyncStatus.SYNCHRONIZATION};get statusIsError(){return this.status===NotSyncStatus.ERROR} }
```

```
src\app\components-modules\not-sync-status-icon\not-sync-status-icon.module.ts
import { NgModule } from '@angular/core';import { CommonModule } from '@angular/common';import { NotSyncStatusIconComponent } from './not-sync-status-icon.component';import { AngularSvgIconModule } from 'angular-svg-icon'@NgModule({ declarations:[NotSyncStatusIconComponent],imports:[CommonModule,AngularSvgIconModule.forRoot()],exports:[NotSyncStatusIconComponent]});export class NotSyncStatusIconModule{ }
```

```
src\app\components-modules\panel-form\panel-form.module.ts
import { NgModule } from '@angular/core';import { CommonModule } from '@angular/common';import { PanelFormDividerComponent } from './panel-form-divider/panel-form-divider.component';import { PanelFormItemComponent } from './panel-form-item/panel-form-item.component';import { AngularSvgIconModule } from 'angular-svg-icon'@NgModule({ declarations:[PanelFormDividerComponent,PanelFormItemComponent],imports:[CommonModule,AngularSvgIconModule],exports:[PanelFormDividerComponent,PanelFormItemComponent]});export class PanelFormModule{ }
```

```
src\app\components-modules\panel-form\panel-form-divider\panel-form-divider.component.ts
import { Component } from '@angular/core'@Component({ selector:'counter-panel-form-divider',templateUrl: './panel-form-divider.component.html' });export class PanelFormDividerComponent{ }
```

```
src\app\components-modules\panel-form\panel-form-item\panel-form-item.component.ts
import { Component, Input, EventEmitter, Output } from '@angular/core'@Component({ selector:'counter-panel-form-item',templateUrl: './panel-form-item.component.html' });export class
```

```
PanelFormItemComponent{ @Input()text:string="@Input()imageSrc:string="@Input()i
imageType:'image'|'svg'='image'@Input()imageClass:string="@Input()disabled:boolean
=false@Output()onClick=new EventEmitter();onButtonClick(){this.onClick.emit()}}
```

```
src\app\components-modules\table-controls\table-controls.component.ts
import{ Component,EventEmitter,Input,Output }from'@angular/core'@Component({ sel
ector:'counter-table-controls',templateUrl:'./table-controls.component.html'});export
class TableControlsComponent{ @Output()delete=new
EventEmitter()@Output()edit=new EventEmitter()@Output()closeEdit=new
EventEmitter()@Input()editActive:boolean=false;onDeleteClick(){this.delete.emit()};o
nEditClick(event:MouseEvent){event.stopPropagation();this.edit.emit()};closeCategory
Edit(){this.closeEdit.emit()}}
```

```
src\app\components-modules\table-controls\table-controls.module.ts
import{ NgModule }from'@angular/core';import{ CommonModule }from'@angular/com
mon';import{ TableControlsComponent }from'./table-
controls.component';import{ AngularSvgIconModule }from'angular-svg-
icon'@NgModule({ declarations:[TableControlsComponent],imports:[CommonModule,
AngularSvgIconModule.forRoot()],exports:[TableControlsComponent]});export class
TableControlsModule{ }
```

```
src\app\directives\clicked-outside-directive.module.ts
import{ NgModule }from'@angular/core';import{ CommonModule }from'@angular/com
mon';import{ ClickedOutsideDirective }from'./clicked-
outside.directive'@NgModule({ declarations:[ClickedOutsideDirective],imports:[Comm
onModule],exports:[ClickedOutsideDirective]});export class
ClickedOutsideDirectiveModule{ }
```

```
src\app\directives\clicked-outside.directive.ts
import{ Directive,ElementRef,EventEmitter,HostListener,Output }from'@angular/core'
@Directive({ selector:'[appClickedOutside]'});export class
ClickedOutsideDirective{ constructor(private
el:ElementRef){ } @Output()clickedOutside=new
EventEmitter()@HostListener('document:click',['$event.target']);public
onClick(target:EventTarget|null){ const
clickedInside=this.el.nativeElement.contains(target);if(!clickedInside){ this.clickedOutsi
de.emit(target)}}
```

```
src\app\directives\keydown-directive.module.ts
import{ NgModule }from'@angular/core';import{ CommonModule }from'@angular/com
mon';import{ KeydownDirective }from'./keydown.directive'@NgModule({ declarations:[
KeydownDirective],imports:[CommonModule],exports:[KeydownDirective]});export
class KeydownDirectiveModule{ }
```

```
src\app\directives\keydown.directive.ts
import { Directive, ElementRef, EventEmitter, HostListener, Output } from '@angular/core'
@Directive({ selector: '[appKeydown]' }); export class
KeydownDirective { constructor(private el: ElementRef) { } @Output() keydownPress = new
EventEmitter() @Output() escapePress = new
EventEmitter() @HostListener('document:keydown', ['$event']); public
onClick(event: KeyboardEvent) { this.keydownPress.emit(event); this.callSpecifyKeysEv
ents(event); } private
callSpecifyKeysEvents(event: KeyboardEvent) { switch(event.key) { case 'Escape': return
this.escapePress.emit(); } } }
```

```
src\app\helpers\index.ts
export * from "
```

```
src\app\helpers\sorted-by-order.helper.ts
export function sortByOrder<T>
extends { order: number } > (array: Array<T> | null): Array<T> | null { if (!array) return
null; return [...array].sort((a, b) => { return a.order > b.order ? 1 : -1 }) }
```

```
src\app\modules\buttons\buttons.module.ts
import { NgModule } from '@angular/core'; import { CommonModule } from '@angular/com
mon'; import { OutlineButtonComponent } from './outline-button/outline-
button.component' @NgModule({ declarations: [OutlineButtonComponent], imports: [Co
mmonModule], exports: [OutlineButtonComponent] }); export class ButtonsModule { }
```

```
src\app\modules\buttons\outline-button\outline-button.component.ts
import { Component, EventEmitter, Output } from '@angular/core' @Component({ selector: '
counter-outline-button', templateUrl: './outline-
button.component.html', styleUrls: ['./outline-button.component.scss'] }); export class
OutlineButtonComponent { @Output() buttonClick = new
EventEmitter(); onClick() { this.buttonClick.emit(); } }
```

```
src\app\modules\categories\categories.module.ts
import { NgModule } from '@angular/core'; import { CommonModule } from '@angular/com
mon'; import { DragDropModule } from '@angular/cdk/drag-
drop'; import { CategoriesPageComponent } from './categories-page/categories-
page.component'; import { AngularSvgIconModule } from 'angular-svg-
icon'; import { FormsModule, ReactiveFormsModule } from '@angular/forms'; import { Click
edOutsideDirectiveModule } from 'src/app/directives/clicked-outside-
directive.module'; import { KeydownDirectiveModule } from 'src/app/directives/keydown-
directive.module'; import { CategorySelectComponent } from './category-select/category-
select.component'; import { ButtonsModule } from '../buttons/buttons.module'; import { Load
StatusButtonModule } from 'src/app/components-modules/load-status-button/load-status-
button.module'; import { CategoriesTableComponent } from './categories-table/categories-
```

```

table.component';import{CategoriesFormComponent}from'./categories-form-add-
new/categories-
form.component';import{NotSyncStatusIconModule}from'../components-
modules/not-sync-status-icon/not-sync-status-
icon.module';import{CategoriesTableColorHeadItemComponent}from'./categories-
table/categories-table-color-head-item/categories-table-color-head-
item.component';import{LottieModule}from'ngx-
lottie';import{TableControlsModule}from'../components-modules/table-
controls/table-
controls.module';import{CategorySelectDropdownComponent}from'./category-
select/category-select-dropdown/category-select-
dropdown.component';import{HighlightPipe}from'./category-
select/highlight.pipe';import{CategoriesTableControlComponent}from'./categories-
table-control/categories-table-
control.component';import{PanelFormModule}from'../components-modules/panel-
form/panel-form.module';exportfunctionplayerFactory(){returnimport('lottie-
web')}@NgModule({declarations:[CategoriesPageComponent,CategorySelectCompone
nt,CategoriesTableComponent,CategoriesFormComponent,CategoriesTableColorHeadIt
emComponent,CategorySelectDropdownComponent,HighlightPipe,CategoriesTableCo
ntrolComponent],imports:[CommonModule,ReactiveFormsModule,FormsModule,Ang
ularSvgIconModule.forRoot(),ClickedOutsideDirectiveModule,KeydownDirectiveMod
ule,ButtonsModule,LoadStatusButtonModule,NotSyncStatusIconModule,LottieModule.
forRoot({player:playerFactory}),TableControlsModule,DragDropModule,PanelFormM
odule],exports:[CategoriesPageComponent,CategorySelectComponent]});exportclass
CategoriesModule{ }

```

```

src\app\modules\categories\categories-form-add-new\categories-form.component.ts
import{Component,OnInit,EventEmitter,Output,Input}from'@angular/core';import{For
mGroup,FormControl}from'@angular/forms';import{Store}from'@ngrx/store';import{
Color}from'src/types/Color';import{sortedByOrder}from'src/app/helpers';import{RootS
tate}from'src/app/store';import{selectColors}from'src/app/store/colors';import{Categori
esBasicSet}from'src/types/ApiInputs'@Component({selector:'counter-categories-
form',templateUrl:'./categories-form.component.html',styleUrls:['./categories-
form.component.scss']});exportclassCategoriesFormComponentimplements
OnInit{@Input()initialFormData:{name:string;comment:string;color:string;dimension:s
tring}|null=null@Input()fromType:'add'|'edit'='add'@Output()onSubmit=new
EventEmitter<CategoriesBasicSet>();colors:Color[]|null=null;formData=new
FormGroup({name:new
FormControl<string>(this.initialFormData?.name||''),comment:new
FormControl<string>(this.initialFormData?.comment||''),color:new
FormControl<string|null>(this.initialFormData?.color||null),dimension:new
FormControl<string>(this.initialFormData?.dimension||'')});ngOnInit(){this.store.select(
selectColors).subscribe(newColors=>{this.colors=newColors});if(this.initialFormData)
{this.formData.setValue(this.initialFormData)};onSubmitAddForm(){const

```



```

valueForSend=this.prepareSubmitData(this.formData.value);if(valueForSend==null){re
turn};this.onSubmit.emit(valueForSend);this.formData.reset()};get
sortedByOrderColors(){return sortedByOrder(this.colors)};private
prepareSubmitData(value:typeof
this.formData.value){if(!value.name||value.comment==null||!value.color||value.dimensi
on==null){return null};const
valueForSend={name:value.name,comment:value.comment,color:value.color,dimensio
n:value.dimension};return valueForSend};get formTypeIsAdd(){return
this.fromType==='add'};get formTypeIsEdit(){return
this.fromType==='edit'};constructor(private store:Store<RootState>){}}

```

```

src\app\modules\categories\categories-page\categories-page.component.ts
import { Component, OnInit } from '@angular/core';import { Store } from '@ngrx/store';import
t { RootState } from 'src/app/store';import { CategoriesActions } from 'src/app/store/categorie
s';import { selectCategoriesState } from 'src/app/store/categories/categories.select';import {
LoadStatus } from 'src/app/store/store.types';import { CategoriesBasicSet } from 'src/types/A
piInputs';@Component({selector:'counter-categories-page',templateUrl:'./categories-
page.component.html'});export class CategoriesPageComponent implements
OnInit{currentStatus:LoadStatus=LoadStatus.NOT_SYNCHRONIZED;public
isAddFormOpened=false;ngOnInit(){this.store.select(selectCategoriesState).subscribe(v
alue=>{this.currentStatus=value});toggleAddForm(){this.isAddFormOpened=!this.isA
ddFormOpened};closeAddForm(){this.isAddFormOpened=false};closeWithCheck(){if(
!this.isAddFormOpened)return;this.closeAddForm()};reloadCategories(force:boolean){
this.store.dispatch(CategoriesActions.load({force:force}))};addNewCategory(data:Cate
goriesBasicSet){this.store.dispatch(CategoriesActions.add(data));this.closeWithCheck()
};constructor(private store:Store<RootState>){}}

```

```

src\app\modules\categories\categories-table\categories-table.component.ts
import { CdkDragDrop,CdkDragEnd,CdkDragStart,moveItemInArray } from '@angular/cd
k/drag-
drop';import { Component,OnInit } from '@angular/core';import { Store } from '@ngrx/store';
import { sortedByOrder } from 'src/app/helpers';import { RootState } from 'src/app/store';imp
ort { selectCategories,CategoriesActions } from 'src/app/store/categories';import { Category
StateItemWithColor,CategorySyncStateItemWithColor } from 'src/app/store/categories/ca
tegories.types'@Component({selector:'counter-categories-
table',templateUrl:'./categories-table.component.html',styleUrls:['./categories-
table.component.scss']});export class CategoriesTableComponent implements
OnInit{categories:CategoryStateItemWithColor[]|null=null;showMenu:string|null=null;
editCategoryId:string|null=null;ngOnInit(){this.store.select(selectCategories).subscribe(
value=>{this.categories=value});get sortedByOrderCategories(){return
sortedByOrder(this.categories)};drop(event:CdkDragDrop<CategorySyncStateItemWit
hColor[],CategorySyncStateItemWithColor[],CategorySyncStateItemWithColor>){if(th
is.sortedByOrderCategories===null)return;if(event.previousIndex===event.currentInde
x)return;const categoryData=event.item.data;const

```

```

previousIndex=categoryData.order;const
currentIndex=this.sortedByOrderCategories[event.currentIndex].order;this.store.dispatch
h(CategoriesActions.reorder({category:categoryData,previousIndex:previousIndex,current
entIndex:currentIndex}));onControlButtonClick(categoryId:string){console.log('onCo
ntrolButtonClick');if(this.showMenu===null||this.showMenu!==categoryId){this.show
Menu=categoryId;return};this.showMenu=null};onFormClickedOutside(event:any,cate
goryId:string){if(this.showMenu!==categoryId){return};this.showMenu=null;this.editC
ategoryId=null};editCategory(currentValue:CategoryStateItemWithColor,editedValue:a
ny){this.editCategoryId=null;this.showMenu=null;this.store.dispatch(CategoriesActions
.update({oldCategory:currentValue,dataForUpdate:editedValue}));setEditCategoryId(
newId:string){if(this.editCategoryId===newId){this.editCategoryId=null;return};this.ed
itCategoryId=newId};deleteCategory(category:CategoryStateItemWithColor){this.store
.dispatch(CategoriesActions.delete(category));constructor(private
store:Store<RootState>){}}

```

```

src\app\modules\categories\categories-table\categories-table-color-head-
item\categories-table-color-head-item.component.ts
import { Component } from '@angular/core';import { AnimationItem } from 'lottie-
web'@Component({ selector: 'counter-categories-table-color-head-
item', templateUrl: './categories-table-color-head-
item.component.html', styleUrls: ['./categories-table-color-head-
item.component.scss'] });export class
CategoriesTableColorHeadItemComponent { animationItem: AnimationItem | null = null; on
Animate(new AnimationItem: AnimationItem): void { if (this.animationItem !== null) return;
this.animationItem = new AnimationItem }; mouseEnter() { this.animationItem?.play() }; mo
useLeave() { this.animationItem?.stop() } }

```

```

src\app\modules\categories\categories-table-control\categories-table-
control.component.ts
import { Component, Input, EventEmitter, Output } from '@angular/core';import { Store } fro
m '@ngrx/store';import { RootState } from 'src/app/store';import { CategoriesActions } from 's
rc/app/store/categories';import { CategoryStateItemWithColor } from 'src/app/store/categor
ies/categories.types';@Component({ selector: 'counter-categories-table-
control', templateUrl: './categories-table-control.component.html', styleUrls: ['./categories-
table-control.component.scss'] });export class
CategoriesTableControlComponent { @Input() category: CategoryStateItemWithColor | nul
l = null @Input() allowShowHint: boolean = true @Output() onClick = new
EventEmitter(); showHint: boolean = false; editCategoryId: string | null = null; deleteCategory(
category: CategoryStateItemWithColor) { this.store.dispatch(CategoriesActions.delete(cat
egory)) }; editCategoryStatus(category: CategoryStateItemWithColor) { if (this.editCategor
yId === null) { this.editCategoryId = category._id; return }; this.editCategoryId = null }; editCa
tegory(currentValue: CategoryStateItemWithColor, editedValue: any) { this.editCategoryId
= null; this.store.dispatch(CategoriesActions.update({ oldCategory: currentValue, dataFor
Update: editedValue })); closeCategoryEdit() { if (this.editCategoryId !== null) { this.editCa

```

```

tegrityId=null}};onMouseEnter(){this.showHint=true};onMouseLeave(){this.showHint=false};onClickInner(){console.log('onClick');this.onClick.emit();this.showHint=false};onMouseDown(){console.log('onMouseDown');this.showHint=false};constructor(private store:Store<RootState>){}}

```

```

src/app/modules/categories/category-select/category-select.component.ts
import { Component, forwardRef, Input, Output, EventEmitter } from '@angular/core'; import
{ ControlValueAccessor, NG_VALUE_ACCESSOR } from '@angular/forms'; import { CategoryStateItemWithColor } from 'src/app/store/categories/categories.types' @Component({ selector: 'counter-category-select', templateUrl: './category-select.component.html', providers: [{ provide: NG_VALUE_ACCESSOR, useExisting: forwardRef(() => CategorySelectComponent), multi: true }] }); export class CategorySelectComponent implements ControlValueAccessor { @Input() categories: CategoryStateItemWithColor[] | null = null @Input() buttonClass: string = ''; isDropDownOpened = false; disabled = false; currentId: string | null = null; currentCategory: CategoryStateItemWithColor | null = null; onChange: any = () => {}; onTouched: any = () => {}; onItemClick(id: string | null) { this.currentId = id; this.onChange(id); this.setCurrentCategory(); this.isDropDownOpened = false; } writeValue(value: string): void { this.currentId = value; this.setCurrentCategory(); } registerOnChange(callback: (_: any) => void): void { this.onChange = callback; } registerOnTouched(callback: (_: any) => void): void { this.onTouched = callback; } setDisabledState?(isDisabled: boolean): void { this.disabled = isDisabled; } setCurrentCategory() { if (this.categories) { this.currentCategory = this.categories.find(category => category._id === this.currentId) || null; } } closeDropDown() { this.isDropDownOpened = false; } toggleDropDown() { this.isDropDownOpened = !this.isDropDownOpened; } }

```

```

src/app/modules/categories/category-select/highlight.pipe.ts
import { Pipe, PipeTransform } from '@angular/core'; import { DomSanitizer } from '@angular/platform-browser' @Pipe({ name: 'highlight' }); export class HighlightPipe implements PipeTransform { constructor(private sanitizer: DomSanitizer) {} transform(value: string, searchValue: string): any { if (!searchValue) return value; const re = new RegExp(searchValue, 'gi'); const match = value.match(re); if (!match) { return value; } const replacedValue = value.replace(re, '<mark>' + match[0] + '</mark>'); return this.sanitizer.bypassSecurityTrustHtml(replacedValue); } }

```

```

src/app/modules/categories/category-select/category-select-dropdown/category-select-dropdown.component.ts
import { Component, EventEmitter, Input, Output } from '@angular/core'; import { Store } from '@ngrx/store'; import { sortByOrder } from 'src/app/helpers'; import { CategoriesActions } from 'src/app/store/categories'; import { CategoryStateItemWithColor } from 'src/app/store/categories/categories.types'; import { CategoriesBasicSet } from 'src/types/ApiInputs'; import { RootState } from 'src/store/rootTypes'; import { HighlightPipe } from '../highlight.pipe' @Component({ selector: 'counter-category-select-dropdown', templateUrl: './category-select-

```

```
select-dropdown.component.html',styleUrls:['./category-select-
dropdown.component.scss'],providers:[HighlightPipe]));export class
CategorySelectDropdownComponent{ @Input()categories:CategoryStateItemWithColor
[]|null=null@Input()currentCategory:CategoryStateItemWithColor|null=null@Output()i
temClick=new EventEmitter<string|null>();searchValue:string="";get
categoriesList(){if(!this.categories)return null;const
searchedCategories=this.categories.filter(item=>item.name.toLocaleLowerCase().includ
es(this.searchValue.toLocaleLowerCase()));if(searchedCategories.length===0)return
null;const categoriesSortedByOrder=sortedByOrder(searchedCategories);return
categoriesSortedByOrder};constructor(private
store:Store<RootState>){};onItemClick(id:string|null){this.itemClick.emit(id)};isAddF
ormOpened:boolean=false;closeAddFormWithCheck(){if(!this.isAddFormOpened)retur
n;this.closeAddForm();toggleAddForm(){this.isAddFormOpened=!this.isAddFormOp
ened};closeAddForm(){this.isAddFormOpened=false};addNewCategory(data:Category
esBasicSet){this.store.dispatch(CategoriesActions.add(data));this.closeAddFormWithC
heck()}}
```

```
src/app/modules/home/home.module.ts
import{NgModule}from'@angular/core';import{CommonModule}from'@angular/com
mon';import{HomePageComponent}from'./home-page/home-
page.component';import{CategoriesModule}from'./categories/categories.module';impo
rt{StatisticModule}from'./statistic/statistic.module';import{FormsModule,ReactiveFo
rmsModule}from'@angular/forms';import{AngularSvgIconModule}from'angular-svg-
icon'@NgModule({declarations:[HomePageComponent],imports:[CommonModule,Cat
egoriesModule,StatisticModule,ReactiveFormsModule,AngularSvgIconModule,Forms
Module],exports:[HomePageComponent]});export class HomeModule{ }
```

```
src/app/modules/home/home-page/home-page.component.ts
import{Component,OnInit}from'@angular/core';import{FormGroup,FormControl}from
'@angular/forms';import{Store}from'@ngrx/store';import{RootState}from'src/app/store
';import{selectCategories}from'src/app/store/categories';import{CategoryStateItemWith
Color}from'src/app/store/categories/categories.types';import{StatisticActions}from'src/
app/store/statistic'@Component({selector:'counter-home-page',templateUrl:'./home-
page.component.html',styleUrls:['./home-page.component.scss']});export class
HomePageComponent implements OnInit{addForm=new FormGroup({count:new
FormControl<number>(0),comment:new FormControl<string>(""),category:new
FormControl<string|null>(null)});categories:CategoryStateItemWithColor[]|null=null;a
dditionalSettingsShow=false;additinalOptions={doNotClearComment:false,doNotClear
Count:false};ngOnInit(){this.store.select(selectCategories).subscribe(value=>{this.cate
gories=value});onSubmit(){const
valueForSend=this.prepareSubmitData(this.addForm.value);if(valueForSend===null){r
eturn};this.store.dispatch(StatisticActions.add(valueForSend));this.addForm.reset({coun
t:this.additinalOptions.doNotClearCount?valueForSend.count:0,comment:this.additinal
Options.doNotClearComment?valueForSend.comment:"",category:valueForSend.categor
```

```

y}});private prepareSubmitData(value:typeof
this.addForm.value){if(value.count==null||value.comment==null||value.category==null)
{return null};const
valueForSend={count:value.count,comment:value.comment,category:value.category,date:new Date(Date.now()).toISOString(),sum:0};return
valueForSend};toggleAdditionalOptionsShow(){this.additionalSettingsShow=!this.additionalSettingsShow};constructor(private store:Store<RootState>){}}

```

```

src\app\modules\statistic\statistic.module.ts
import { NgModule } from '@angular/core';import { CommonModule } from '@angular/common';import { StatisticComponent } from './statistic-page/statistic-page.component';import { StatisticChartComponent } from './statistic-chart/statistic-chart.component';import { AngularSvgIconModule } from 'angular-svg-icon';import { StatisticLogComponent } from './statistic-log/statistic-log.component';import { LoadStatusButtonModule } from 'src/app/components-modules/load-status-button/load-status-button.module';import { NotSyncStatusIconModule } from 'src/app/components-modules/not-sync-status-icon/not-sync-status-icon.module';import { TableControlsModule } from 'src/app/components-modules/table-controls/table-controls.module';import { StatisticFormComponent } from './statistic-form/statistic-form.component';import { ReactiveFormsModule } from '@angular/forms';import { CategoriesModule } from 'src/app/categories/categories.module';import { ClickedOutsideDirectiveModule } from 'src/app/directives/clicked-outside-directive.module';import { KeydownDirectiveModule } from 'src/app/directives/keydown-directive.module';@NgModule({declarations:[StatisticComponent,StatisticChartComponent,StatisticLogComponent,StatisticFormComponent],imports:[CommonModule,ReactiveFormsModule,AngularSvgIconModule.forRoot(),LoadStatusButtonModule,NotSyncStatusIconModule,TableControlsModule,CategoriesModule,ClickedOutsideDirectiveModule,KeydownDirectiveModule],exports:[StatisticComponent,StatisticLogComponent]});export class StatisticModule {}

```

```

src\app\modules\statistic\helpers\sort-by-date.helper.ts
export function sortByDate<T extends { date:string }>(a:T,b:T):number{return new Date(a.date).getTime()-new Date(b.date).getTime()}

```

```

src\app\modules\statistic\statistic-chart\statistic-chart.component.ts
import { Component,Input,OnInit,SimpleChanges } from '@angular/core';import Chart from 'chart.js/auto';import 'chartjs-adapter-moment';import { StatisticStateItemWithCategory } from 'src/app/store/statistic/statistic.types';@Component({selector:'counter-statistic-chart',templateUrl:'./statistic-chart.component.html',styleUrls:['./statistic-chart.component.scss']});export class StatisticChartComponent implements
OnInit{ @Input()statistics:StatisticStateItemWithCategory[]=[];currentChartType:'day'|'r

```

```

record='day';private
chart:Chart;ngOnChanges(changes:SimpleChanges){console.log(changes);if(!changes['
statistics'].firstChange){this.chart.data=this.chartData(this.statistics,this.currentChartTy
pe);this.chart.update()}};changeChartType(){if(this.currentChartType==='day'){this.cur
rentChartType='record'}else{this.currentChartType='day'};this.chart.data=this.chartDat
a(this.statistics,this.currentChartType);console.log(this.chart.data);this.chart.update()};c
hartData(statistics:StatisticStateItemWithCategory[],type:'day'|'record'='day'){const
temp:{categoryName:string;data:Array<any>;colorHEX:string}[]=[];statistics.forEach(r
ecord=>{if(!record.category)return;const categoryName=record.category.name;let
finedRecord=temp.find(item=>item.categoryName===categoryName);if(finedReco
rd===undefined){const
index=temp.push({categoryName:categoryName,colorHEX:record.category.color.color
HEX,data:[]});finedRecord=temp[index-1]};if(type==='day'){const date=new
Date(new Date(record.date).toDateString()).getTime();const
finedRecordData=finedRecord.data.find(item=>item.x===date);if(finedRecordData
===undefined){finedRecord.data.push({x:date,y:record.count});return};finedRecord
Data.y+=record.count;return};if(type==='record'){const date=new
Date(record.date).getTime();finedRecord.data.push({x:date,y:record.count});return}});
const
datasets=temp.map(item=>{return{label:item.categoryName,data:item.data,tension:0.4,
borderColor:item.colorHEX,backgroundColor:item.colorHEX}});return{datasets}};ng
OnInit():void{this.chart=new Chart('statistic-chart-
canvas',{type:'line',data:this.chartData(this.statistics,this.currentChartType),options:{res
ponsive:true,spanGaps:true,interaction:{intersect:false},scales:{x:{type:'time',time:{unit
:'day'}}}},animation:{duration:400},animations:{y:{duration:0}}}})}

```

```

src\app\modules\statistic\statistic-form\statistic-form.component.ts
import{Component,Input,Output,EventEmitter}from'@angular/core';import{FormContr
ol,FormGroup}from'@angular/forms';import{Store}from'@ngrx/store';import{RootStat
e}from'src/app/store';import{CategoryStateItemWithColor}from'src/app/store/categorie
s/categories.types';import{AddStatisticInputs}from'../types/ApiInputs';import{sele
ctCategories}from'../store/categories/categories.select'@Component({selector:'coun
ter-statistic-form',templateUrl:'./statistic-form.component.html',styleUrls:['./statistic-
form.component.scss']}));export class
StatisticFormComponent{ @Input()initialFormData:{date:string;count:number;comment
:string;category:string}|null=null @Input()fromType:'add'|'edit'='add' @Output()onSubmi
t=new EventEmitter<AddStatisticInputs>();formData=new FormGroup({date:new
FormControl<string>(''),count:new FormControl<number>(0),comment:new
FormControl<string>(''),category:new
FormControl<string|null>(null)});categories:CategoryStateItemWithColor[]|null=null;lo
calSelectCategoriesDropDownStatus:boolean=false;ngOnInit(){this.store.select(selectC
ategories).subscribe(value=>{this.categories=value});if(!this.initialFormData){throw
new Error('initialFormData on edit stastistic
required!')};this.initialFormData.date=this.formatDateToDateTimeLocalInput(this.initia

```

```

IFormData.date);this.formData.setValue(this.initialFormData));private
formatDateToDateTimeLocalInput(input:string){const inputAsDate=new
Date(input);const dateWithTimezoneOffset=new Date(inputAsDate.getTime()-
inputAsDate.getTimezoneOffset()*60000);const
result=dateWithTimezoneOffset.toISOString().slice(0,-5);return
result};onSubmitAddForm(){const
valueForSend=this.prepareSubmitData(this.formData.value);if(valueForSend===null){re
turn};this.onSubmit.emit(valueForSend)};private prepareSubmitData(value:typeof
this.formData.value){if(!value.count||value.comment===null||!value.date||!value.category
){return null};const
valueForSend={count:value.count,comment:value.comment,category:value.category,date:
new Date(value.date).toISOString(),summ:-1};return valueForSend};get
formTypeIsAdd(){return this.fromType==='add'};get formTypeIsEdit(){return
this.fromType==='edit'};testClick(event:MouseEvent){console.log('select
form:',event.target);console.log('select
form:',event.currentTarget);if(this.localSelectCategoriesDropDownStatus){};event.stop
Propagation()};setLocalSelectCategoriesDropDownStatus(value:boolean){this.localSele
ctCategoriesDropDownStatus=value};constructor(private store:Store<RootState>){}}

```

```

src/app/modules/statistic/statistic-log/statistic-log.component.ts
import { Component } from '@angular/core';import { Store } from '@ngrx/store';import { Root
State } from 'src/app/store';import { selectNotSyncStatisticWithCategory } from 'src/app/stor
e/statistic/statistic.select';import { StatisticNotSyncStateItemWithCategory } from 'src/app/
store/statistic/statistic.types';import { NotSyncStatus } from 'src/app/store/store.types';type
LocalStateItem=StatisticNotSyncStateItemWithCategory&{isSuccess?:never;isSuccess
Status?:never};type
LocalStateItemNotSync=StatisticNotSyncStateItemWithCategory&{isSuccess:boolean;
isSuccessStatus:string|null}@Component({ selector:'counter-statistic-
log',templateUrl:'./statistic-log.component.html',styleUrls:['./statistic-
log.component.scss']});export class StatisticLogComponent{constructor(private
store:Store<RootState>){};notSyncStatistic:LocalStateItem[]|null=null;notSyncStatistic
SuccessesItems:LocalStateItemNotSync[]=[];statusColors={ [NotSyncStatus.NOT_SYN
CHRONIZED]:'#d4d4d8',[NotSyncStatus.SYNCHRONIZATION]:'#fde68a',[NotSyncSt
atus.ERROR]:'#fca5a5',success:'#86efac'};ngOnInit(){this.store.select(selectNotSyncSt
atisticWithCategory).subscribe(value=>{this.updateNoSyncLocalState(value)});privat
e
updateNoSyncLocalState(newState:LocalStateItem[]){if(!this.notSyncStatistic){this.not
SyncStatistic=[...newState];return};const
deletedItems=this.notSyncStatistic.map(item=>{const
newStateItem=newState.find(data=>data._id===item._id);if(!newStateItem){return
item};return null});const filtered=deletedItems.filter(v=>v!==null)as
LocalStateItem[];this.addNewSuccededItems(filtered);this.notSyncStatistic=[];this.notS
yncStatistic.push(...newState)};private
addNewSuccededItems=(items:LocalStateItem[])=>{const

```

```

newItems:LocalStateItemNotSync[]=items.map(item=>({ ...item,isSuccess:true,isSuccessStatus:null}));newItems.forEach(item=>{ setTimeout(()=>{ item.isSuccessStatus='deleting';setTimeout(()=>{ this.deleteNewSuccededItem(item._id)},500)},2000)});this.notSyncStatisticSuccessesItems=[...this.notSyncStatisticSuccessesItems,...newItems];private deleteNewSuccededItem=(id:string)=>{ this.notSyncStatisticSuccessesItems=this.notSyncStatisticSuccessesItems.filter(item=>item._id!==id)};get stasticItems(): (LocalStateItem|LocalStateItemNotSync)[] { return [...this.notSyncStatisticSuccessesItems,...(this.notSyncStatistic||[])] }

```

```

src\app\modules\statistic\statistic-page\statistic-page.component.ts
import { Component,OnInit } from '@angular/core';import { Store } from '@ngrx/store';import { selectStatistic,StatisticActions } from 'src/app/store/statistic';import { RootState } from 'src/app/store/rootTypes';import { LoadStatus,NotSyncStatus } from 'src/app/store/store.types';import { selectStatisticStatus } from 'src/app/store/statistic/statistic.select';import { StatisticStateItemWithCategory } from 'src/app/store/statistic/statistic.types';import { sortByDate } from '../helpers/sort-by-date.helper'@Component({ selector:'counter-statistic-page',templateUrl:'./statistic-page.component.html',styleUrls:['./statistic-page.component.scss']});export class StatisticComponent implements OnInit { statistics:StatisticStateItemWithCategory[]|null=null;currentStatus:LoadStatus|null=null;editStatisticRecordId:string|null=null;ngOnInit() { this.store.select(selectStatistic).subscribe(newStatistic=>{ this.statistics=newStatistic.sort(sortByDate)});this.store.select(selectStatisticStatus).subscribe(value=>{ this.currentStatus=value});constructor(private store:Store<RootState>){};reloadStatistic(force:boolean){ this.store.dispatch(StatisticActions.load({ force:force}));checkStatusIs(status:NotSyncStatus|undefined,value:'not-synchronized'|'synchronization'|'error'){ if(!status)return;switch(value){ case'not-synchronized':return status===NotSyncStatus.NOT_SYNCHRONIZED;case'synchronization':return status===NotSyncStatus.SYNCHRONIZATION;case'error':return status===NotSyncStatus.ERROR}};deleteStatisticRecord(statisticRecord:StatisticStateItemWithCategory){ this.store.dispatch(StatisticActions.delete(statisticRecord));closeStatisticEdit(){ this.editStatisticRecordId=null};editStatisticStatus(category:StatisticStateItemWithCategory){ if(this.editStatisticRecordId===null){ this.editStatisticRecordId=category._id;return};this.editStatisticRecordId=null};editStatistic(currentValue:StatisticStateItemWithCategory,editedValue:any){ this.editStatisticRecordId=null;this.store.dispatch(StatisticActions.update({ oldStatistic:currentValue,dataForUpdate:editedValue}))}

```

```

src\app\modules\user\user.module.ts
import { NgModule } from '@angular/core';import { CommonModule } from '@angular/common';import { UserPanelComponent } from './user-panel/user-panel.component';import { UserPanelPlugComponent } from './user-panel/user-panel-plug/user-panel-plug.component';import { UserPanelOpenButtonComponent } from './user-panel/user-panel-open-button/user-panel-open-button.component';import { UserPanelFormComponent } from './user-panel/user-panel-

```



```

form/user-panel-
form.component';import{ ClickedOutsideDirectiveModule }from'src/app/directives/click
ed-outside-
directive.module';import{ KeydownDirectiveModule }from'src/app/directives/keydown-
directive.module';import{ PanelFormModule }from'src/app/components-modules/panel-
form/panel-
form.module'@NgModule({ declarations:[UserPanelComponent,UserPanelPlugCompon
ent,UserPanelOpenButtonComponent,UserPanelFormComponent],imports:[CommonM
odule,ClickedOutsideDirectiveModule,KeydownDirectiveModule,PanelFormModule],e
xports:[UserPanelComponent]});export class UserModule{ }

```

```

src\app\modules\user\user-panel\user-panel.component.ts
import{ Component,Input }from'@angular/core';import{ User }from'src/types/User';import
t{ AuthService }from'src/app/services/auth-
guard.service'@Component({ selector:'user-panel',templateUrl:'./user-
panel.component.html'});export class UserPanelComponent{ constructor(private
authGuard:AuthService){ } @Input()userInfo:User|null=null;isOpened=false;priva
te toggleDropdownOpened(){ this.isOpened=!this.isOpened };private
closeDropdown(){ this.isOpened=false };onBadgeClick(){ this.toggleDropdownOpened()
};onExitClick(){ this.authGuard.unauthorize();this.closeDropdown() };closeWithChecks(
){ if(this.userInfo===null)return;if(this.isOpened===false)return;this.closeDropdown()
}
}

```

```

src\app\modules\user\user-panel\user-panel-form\user-panel-form.component.ts
import{ Component,EventEmitter,Input,Output }from'@angular/core'@Component({ sel
ector:'user-panel-form',templateUrl:'./user-panel-form.component.html'});export class
UserPanelFormComponent{ @Output()exitClick=new
EventEmitter();onExitClick(){ this.exitClick.emit() } }

```

```

src\app\modules\user\user-panel\user-panel-open-button\user-panel-open-
button.component.ts
import{ Component,EventEmitter,Input,Output }from'@angular/core';import{ User }from'
src/types/User'@Component({ selector:'user-panel-open-button',templateUrl:'./user-
panel-open-button.component.html'});export class
UserPanelOpenButtonComponent{ @Input()userInfo:User|null=null @Input()isOpened:
boolean=false @Output()buttonClick=new
EventEmitter();onButtonClick(){ this.buttonClick.emit() } }

```

```

src\app\modules\user\user-panel\user-panel-plug\user-panel-plug.component.ts
import{ Component }from'@angular/core'@Component({ selector:'user-panel-
plug',templateUrl:'./user-panel-plug.component.html'});export class
UserPanelPlugComponent{ }

```

```

src\app\pages\authorization\authorization.component.ts

```

```
import { Component } from '@angular/core'
@Component({ selector: 'app-authorization', templateUrl: './authorization.component.html' });
export class AuthorizationComponent { }
```

```
src\app\pages\perf\perf.component.ts
import { Component, OnInit } from '@angular/core';
import { Color } from 'src/types/Color';
import { ColorsActions, selectColors } from 'src/app/store/colors';
import { Store } from '@ngrx/store';
import { RootState } from 'src/app/store';
@Component({ selector: 'app-perf', templateUrl: './perf.component.html' });
export class PerfComponent implements OnInit {
  colors: Color[] = [];
  constructor(private store: Store<RootState>) {
    store.select(selectColors).subscribe(newColors => { this.colors = newColors });
  }
  ngOnInit() { this.store.dispatch(ColorsActions.loadColors()); }
}
```

```
src\app\resolvers\category.resolver.ts
import { Injectable } from '@angular/core';
import { Resolve, RouterStateSnapshot, ActivatedRouteSnapshot } from '@angular/router';
import { Store } from '@ngrx/store';
import { RootState } from '../store/rootTypes';
import { CategoriesActions } from '../store/categories/categories.actions';
@Injectable({ providedIn: 'root' });
export class CategoryResolver implements Resolve<boolean> {
  resolve(route: ActivatedRouteSnapshot, state: RouterStateSnapshot): boolean {
    this.store.dispatch(CategoriesActions.load());
    return true;
  }
  constructor(private store: Store<RootState>) { }
}
```

```
src\app\resolvers\color.resolver.ts
import { Injectable } from '@angular/core';
import { Resolve, RouterStateSnapshot, ActivatedRouteSnapshot } from '@angular/router';
import { Store } from '@ngrx/store';
import { RootState } from '../store/rootTypes';
import { ColorsActions } from '../store/colors';
@Injectable({ providedIn: 'root' });
export class ColorResolver implements Resolve<boolean> {
  resolve(route: ActivatedRouteSnapshot, state: RouterStateSnapshot): boolean {
    this.store.dispatch(ColorsActions.loadColors());
    return true;
  }
  constructor(private store: Store<RootState>) { }
}
```

```
src\app\resolvers\statistic.resolver.ts
import { Injectable } from '@angular/core';
import { Resolve, RouterStateSnapshot, ActivatedRouteSnapshot } from '@angular/router';
import { Store } from '@ngrx/store';
import { RootState } from '../store/rootTypes';
import { StatisticActions } from '../store/statistic';
@Injectable({ providedIn: 'root' });
export class StatisticResolver implements Resolve<boolean> {
  resolve(route: ActivatedRouteSnapshot, state: RouterStateSnapshot): boolean {
    this.store.dispatch(StatisticActions.load());
    return true;
  }
  constructor(private store: Store<RootState>) { }
}
```

```
src\app\services\api.service.ts
import { Injectable } from '@angular/core';
import { HttpClient, HttpHeaders } from '@angular/common/http';
import { User } from '../types/User';
import { Category } from 'src/types/Category';
import { Statistic } from '../types/Statistic';
import { Color } from 'src/types/Color';
import { 
```

```

ort{environment}from'src/environments/environment';import*as ApiInputs
from'src/types/ApiInputs';const
API_BASE_URL=environment.API_HOST@Injectable({providedIn:'root'});export
class ApiService{initialize(){const
initialize=this.http.get<ApiInputs.InitializeFailed|ApiInputs.InitializeSuccess>(`${API_
BASE_URL}/initialize`,{withCredentials:true,responseType:'json'});return
initialize};authorization(JWTGoogleAuthorizationToken:string){return
this.http.post<User>(`${API_BASE_URL}/authorization`,null,{withCredentials:true,re
sponseType:'json',headers:new
HttpHeaders({Authorization:JWTGoogleAuthorizationToken}}));logout(){return
this.http.post<{status:'ok'|unknown}>(`${API_BASE_URL}/authorization/logout`,null,
{withCredentials:true,responseType:'json'});getAllStatisticRecords(){return
this.http.get<Statistic[]>(`${API_BASE_URL}/statistic/all`,{withCredentials:true,resp
onseType:'json'});addStatisticRecord(data:ApiInputs.AddStatisticInputs){return
this.http.post<Statistic>(`${API_BASE_URL}/statistic/add`,data,{withCredentials:true
});deleteStatistic(id:string){return
this.http.delete(`${API_BASE_URL}/statistic/${id}`,{withCredentials:true});updateSt
atistic(id:string,data:ApiInputs.AddStatisticInputs){return
this.http.put<Statistic>(`${API_BASE_URL}/statistic/${id}`,data,{withCredentials:true
});getAllColors(){return
this.http.get<Color[]>(`${API_BASE_URL}/color/all`,{withCredentials:true});getAll
Categories(){return
this.http.get<Category[]>(`${API_BASE_URL}/category/all`,{withCredentials:true,res
ponseType:'json'});addCategory(data:ApiInputs.CategoriesBasicSet){return
this.http.post<Category>(`${API_BASE_URL}/category/add`,data,{withCredentials:tru
e});deleteCategory(id:string){return
this.http.delete(`${API_BASE_URL}/category/${id}`,{withCredentials:true});update
Category(id:string,data:ApiInputs.CategoriesBasicSet){return
this.http.put<Category>(`${API_BASE_URL}/category/${id}`,data,{withCredentials:tr
ue});reorderCategory(data:ApiInputs.ReorderCategoryData){return
this.http.put<ApiInputs.ReorderCategoryReturnData>(`${API_BASE_URL}/category/r
eorder`,data,{withCredentials:true});constructor(private http:HttpClient){}}

```

```

src\app\services\auth-guard.service.ts
import{Injectable}from'@angular/core';import{ActivatedRouteSnapshot,CanActivate,R
outer,RouterStateSnapshot}from'@angular/router';import{Observable,ReplaySubject,fi
rst}from'rxjs';import{ApiService}from'./api.service';import{User}from'src/types/User';i
mport*as ApiInputs
from'src/types/ApiInputs';import{SocialAuthService,SocialUser}from'@abacritt/angula
rx-social-login';import{GoogleLoginProvider}from'@abacritt/angularx-social-
login'@Injectable({providedIn:'root'});export class AuthGuardService implements
CanActivate{public authGuardData=new
ReplaySubject<ApiInputs.InitializeFailed|User>(1);private
socialAuthServiceUser:SocialUser|null=null;constructor(private router:Router,private

```

```

apiService: ApiService, private
socialAuthService: SocialAuthService) { this.apiService.initialize().subscribe(value => { this
.isAuthGuardData.next(value) }); this.socialAuthService.authState.subscribe(user => { this
.socialAuthService.user = user; if (user === null) return; this.apiService.authorization(user.id
Token).subscribe(authorizationValue => { this.authorize(authorizationValue) })); canActivate(
route: ActivatedRouteSnapshot, state: RouterStateSnapshot) { return new
Observable<boolean>(subscriber => { this.authGuardData.pipe(first()).subscribe(value =>
{ console.log(value); if (route.url.toString() === 'authorization') { if (value.authorized === true)
{ this.router.navigate(['/']); subscriber.next(false); return }; subscriber.next(true); return };
if (value.authorized === true) { subscriber.next(true); return }; subscriber.next(false); if (this.r
outer.url !== '/authorization') { console.log('%c ', 'background: no-repeat url(https;;
);
this.router.navigate(['/authorization']);
});
});
});
authorize(data: User) {
this.authGuardData.next(data);
this.router.navigate(['/']);
};
unauthorize() {
this.apiService.logout().subscribe(value => {
if (value.status !== 'ok') {
throw new Error('Log
out error! Response /logout: ${value}`);
};
if
(this.socialAuthService.user) {
this.socialAuthService.signOut();
};
this.authGuardData.next({ authorized: false });
this.router.navigate(['/authorization'])
}
}
}

```

src\app\store\index.ts

```

export { default as StoreReducers } from './rootReducers';
export { default as StoreEffects } from './rootEffects';
export * from "

```

src\app\store\rootEffects.ts

```

import { CategoriesEffects } from './categories/categories.effects';
import { ColorEffects } from './colors/colors.effects';
import { StatisticEffects } from './statistic/statistic.effects';
const effects = [ColorEffects, CategoriesEffects, StatisticEffects];
export default effects

```

src\app\store\rootReducers.ts

```

import { ActionReducerMap } from '@ngrx/store';
import { RootState } from './rootTypes';
import { colorsReducer } from './colors';
import { categoriesReducer } from './categories/sync';
import { categoriesNotSyncReducer } from './categories/not-sync';
import { statisticReducer } from './statistic/sync';
import { statisticNotSyncReducer } from './statistic/not-sync';
import { categoriesStatusReducer } from './categories/status';
import { statisticStatusReducer } from './statistic/status';
const reducers: ActionReducerMap<RootState> = {
  colors: colorsReducer,
  categories: categoriesReducer,
  statistic: statisticReducer,
  notSyncCategories: categoriesNotSyncReducer,
  notSyncStatistic: statisticNotSyncReducer,
  categoriesStatus: categoriesStatusReducer,
  statisticStatus: statisticStatusReducer
};
export default reducers

```

```
src/app/store/rootTypes.ts
import { ColorState } from './colors'; import { StatisticTypes } from './statistic'; import { CategoriesTypes } from './categories'; export interface
RootState { colors: ColorState; categories: CategoriesTypes.SyncState; statistic: StatisticTypes.SyncState; notSyncCategories: CategoriesTypes.NotSyncState; notSyncStatistic: StatisticTypes.NotSyncState; categoriesStatus: CategoriesTypes.StatusTypes.StatusState; statisticStatus: StatisticTypes.StatusTypes.StatusState }
```

```
src/app/store/store.types.ts
export enum LoadStatus { NOT_SYNCHRONIZED='not-synchronized', SYNCHRONIZATION='synchronization', ERROR='error', SYNCHRONIZED='synchronized' }; export enum NotSyncStatus { NOT_SYNCHRONIZED='not-synchronized', SYNCHRONIZATION='synchronization', ERROR='error' }; export enum NotSyncAction { CREATED='created', CHANGED='changed', DELETED='deleted' }; export interface
NotSyncStateItemBase { action: NotSyncAction; status: NotSyncStatus }; export interface SyncStateItemBase { action?: never; status?: never }
```

```
src/app/store/categories/categories.actions.ts
import { createActionGroup, props } from '@ngrx/store'; import { CategoriesBasicSet } from 'src/types/ApiInputs'; import { NotSyncStateItem, CategoryStateItemWithColor, CategorySyncStateItemWithColor } from './categories.types'; export const
CategoriesActions = createActionGroup({ source: 'Categories', events: { load: (props: { force: boolean }) => { force: false } } => props, add: (category: CategoriesBasicSet) => category, delete: (category: CategoryStateItemWithColor) => category, update: props < { oldCategory: CategoryStateItemWithColor; dataForUpdate: CategoriesBasicSet } > (), reorder: props < { category: CategorySyncStateItemWithColor; previousIndex: number; currentIndex: number } > (), addEffect: (category: NotSyncStateItem) => category, deleteEffect: (category: NotSyncStateItem) => category, updateEffect: (category: NotSyncStateItem) => category, reorderEffect: props < { category: NotSyncStateItem; previousIndex: number; currentIndex: number } > () } })
```

```
src/app/store/categories/categories.config.ts
export const NOT_SYNC_ID_TAG = "
```

```
src/app/store/categories/categories.effects.ts
import { Injectable } from '@angular/core'; import { Actions, createEffect, ofType } from '@ngrx/effects'; import { select, Store } from '@ngrx/store'; import { EMPTY, of } from 'rxjs'; import { map, switchMap, exhaustMap, catchError, withLatestFrom, mergeMap } from 'rxjs/operators'; import { ApiService } from 'src/app/services/api.service'; import { CategoriesActions } from './categories.actions'; import { CategoriesNotSyncActions } from './not-sync/categories-not-sync.actions'; import { CategoriesSyncActions } from './sync/categories-sync.actions'; import { RootState } from '../rootTypes'; import { NotSyncHelpers } from './not-sync'; import { CategoriesStatusActions, CategoriesStatusTypes } from './status'; import { StatisticActions } from '../statistic/statistic.actions'; import { NotSyncStateItem } from './categories-not-sync.types';
```

```

s.types';import { NotSyncStatus } from '../store.types';import { categoryStateItemWithColor
ToDefault } from './helpers/category-state-item-with-color-to-
default.helper'@Injectable();export class
CategoriesEffects { add$=createEffect(()=>this.actions$.pipe(ofType(CategoriesActions.
add),exhaustMap(categoryForAdd=>{ const
categoryAsNotSyncStateItem:NotSyncStateItem=NotSyncHelpers.changeAddCategory
ValueToStoreItem(categoryForAdd);return
of((CategoriesNotSyncActions.add(categoryAsNotSyncStateItem),CategoriesActions.ad
deffect(categoryAsNotSyncStateItem)))));update$=createEffect(()=>this.actions$.pipe
(ofType(CategoriesActions.update),exhaustMap(categoryForUpdate=>{ const
oldCategory=categoryStateItemWithColorToDefault(categoryForUpdate.oldCategory);i
f(oldCategory.status){ return[CategoriesNotSyncActions.update({ oldCategory:oldCateg
ory,dataForUpdate:categoryForUpdate.dataForUpdate })]);const
oldCategoryAsNotSyncStateItem:NotSyncStateItem=NotSyncHelpers.changeUpdateCa
tegorValueToStoreItem(oldCategory,categoryForUpdate.dataForUpdate);return
of((CategoriesNotSyncActions.add(oldCategoryAsNotSyncStateItem),CategoriesSyncA
ctions.delete(oldCategory),CategoriesActions.updateeffect(oldCategoryAsNotSyncState
Item)))));reorder$=createEffect(()=>this.actions$.pipe(ofType(CategoriesActions.reor
der),exhaustMap(({ category,previousIndex,currentIndex })=>{ const
categoryAsStateItem=categoryStateItemWithColorToDefault(category);if(categoryAsSt
ateItem.status){ return of() };const
oldCategoryAsNotSyncStateItem:NotSyncStateItem={ ...NotSyncHelpers.changeUpdate
CategoryValueToStoreItem(categoryAsStateItem,{ }),order:previousIndex<currentIndex
?currentIndex+1:currentIndex };return
of((CategoriesNotSyncActions.add(oldCategoryAsNotSyncStateItem),CategoriesSyncA
ctions.delete(categoryAsStateItem),CategoriesActions.reordereffect({ category:oldCateg
oryAsNotSyncStateItem,previousIndex,currentIndex })))));delete$=createEffect(()=>th
is.actions$.pipe(ofType(CategoriesActions.delete),exhaustMap(categoryForDeleteInput
=>{ const
categoryForDelete=categoryStateItemWithColorToDefault(categoryForDeleteInput);if(
categoryForDelete.status){ return
of((CategoriesNotSyncActions.delete(categoryForDelete)) );const
categoryAsNotSyncStateItem=NotSyncHelpers.changeDeleteCategoryValueToStoreIt
em(categoryForDelete);return
of((CategoriesNotSyncActions.add(categoryAsNotSyncStateItem),CategoriesSyncActio
ns.delete(categoryForDelete),CategoriesActions.deleteeffect(categoryAsNotSyncStateIt
em)))));loadCategories$=createEffect(()=>this.actions$.pipe(ofType(CategoriesAction
s.load),withLatestFrom(this.store.pipe(select('categories'))),exhaustMap(([params,cate
goriesValue])=>{ if(categoriesValue.length===0||params.force){ this.store.dispatch(Cate
goriesStatusActions.set({ status:CategoriesStatusTypes.StatusState.SYNCHRONIZATIO
N }));return
this.api.getAllCategories().pipe(mergeMap(value=>[CategoriesSyncActions.set({ catego
ries:value }),CategoriesStatusActions.set({ status:CategoriesStatusTypes.StatusState.SY
NCHRONIZED })]),catchError(()=>{ CategoriesStatusActions.set({ status:CategoriesSta

```

```

tusTypes.StatusState.ERROR});return EMPTY}));return
EMPTY}));addCategory$=createEffect(()=>this.actions$.pipe(ofType(CategoriesActio
ns.addeffect),switchMap(inputCategory=>{this.store.dispatch(CategoriesNotSyncActio
ns.changestatus({status:NotSyncStatus.SYNCHRONIZATION,category:inputCategory
}));this.store.dispatch(CategoriesStatusActions.set({status:CategoriesStatusTypes.Status
State.SYNCHRONIZATION}));return
this.api.addCategory(inputCategory).pipe(switchMap(resultCategory=>[CategoriesSync
Actions.add({category:resultCategory}),CategoriesStatusActions.set({status:Categories
StatusTypes.StatusState.SYNCHRONIZED}),CategoriesNotSyncActions.delete(inputC
ategory)]),catchError(()=>{this.store.dispatch(CategoriesNotSyncActions.changestatus(
{status:NotSyncStatus.ERROR,category:inputCategory}));this.store.dispatch(Categorie
sStatusActions.set({status:CategoriesStatusTypes.StatusState.ERROR}));return
EMPTY})))));deleteCategory$=createEffect(()=>this.actions$.pipe(ofType(Categories
Actions.deleteeffect),switchMap(inputCategory=>{this.store.dispatch(CategoriesNotSy
ncActions.changestatus({status:NotSyncStatus.SYNCHRONIZATION,category:inputC
ategory}));this.store.dispatch(CategoriesStatusActions.set({status:CategoriesStatusType
s.StatusState.SYNCHRONIZATION}));return
this.api.deleteCategory(inputCategory._id).pipe(switchMap(()=>[CategoriesStatusActio
ns.set({status:CategoriesStatusTypes.StatusState.SYNCHRONIZED}),CategoriesNotSy
ncActions.delete(inputCategory)]),catchError(()=>{this.store.dispatch(CategoriesNotSy
ncActions.changestatus({status:NotSyncStatus.ERROR,category:inputCategory}));this.
store.dispatch(CategoriesStatusActions.set({status:CategoriesStatusTypes.StatusState.E
RROR}));return
EMPTY})))));updateCategory$=createEffect(()=>this.actions$.pipe(ofType(Categorie
sActions.updateeffect),switchMap(inputCategory=>{this.store.dispatch(CategoriesNotS
yncActions.changestatus({status:NotSyncStatus.SYNCHRONIZATION,category:input
Category}));this.store.dispatch(CategoriesStatusActions.set({status:CategoriesStatusTy
pes.StatusState.SYNCHRONIZATION}));return
this.api.updateCategory(inputCategory._id,inputCategory).pipe(switchMap(resultCateg
ory=>[CategoriesSyncActions.add({category:resultCategory}),CategoriesStatusActions
.set({status:CategoriesStatusTypes.StatusState.SYNCHRONIZED}),CategoriesNotSyn
cActions.delete(inputCategory)]),catchError(()=>{this.store.dispatch(CategoriesNotSyn
cActions.changestatus({status:NotSyncStatus.ERROR,category:inputCategory}));this.st
ore.dispatch(CategoriesStatusActions.set({status:CategoriesStatusTypes.StatusState.ER
ROR}));return
EMPTY})))));reorderCategory$=createEffect(()=>this.actions$.pipe(ofType(Categorie
sActions.reordereffect),switchMap(({category,previousIndex,currentIndex})=>{this.sto
re.dispatch(CategoriesNotSyncActions.changestatus({status:NotSyncStatus.SYNCHRO
NIZATION,category:category}));this.store.dispatch(CategoriesStatusActions.set({statu
s:CategoriesStatusTypes.StatusState.SYNCHRONIZATION}));console.log('data for
reorder:',{categoryId:category._id,previousIndex,currentIndex});return
this.api.reorderCategory({categoryId:category._id,previousIndex,currentIndex}).pipe(s
witchMap(resultValue=>of(...resultValue.map(value=>{if(value.categoryId===categor
y._id){return

```

```
CategoriesSyncActions.add({ category: { ...category, order: value.currentIndex, status: undefined, action: undefined } }));return
CategoriesSyncActions.orderupdate({ data: value })),CategoriesStatusActions.set({ status:CategoriesStatusTypes.StatusState.SYNCHRONIZED}),CategoriesNotSyncActions.delete(category))),catchError(()=>{this.store.dispatch(CategoriesNotSyncActions.change status({ status:NotSyncStatus.ERROR,category:category }));this.store.dispatch(CategoriesStatusActions.set({ status:CategoriesStatusTypes.StatusState.ERROR }));return EMPTY })))));constructor(private actions$:Actions,private api:ApiService,private store:Store<RootState>){ }
```

```
src\app\store\categories\categories.select.ts
import { RootState } from '../rootTypes';import { CategoryStateItemWithColor } from './categories.types';import { changeCategoryIdToColorObject } from './helpers';import { CategoriesStatusTypes } from './status';export const
selectCategories=(state:RootState):CategoryStateItemWithColor[]=>{ return
changeCategoryIdToColorObject([...state.categories,...state.notSyncCategories],state.colors)};export const
selectCategoriesState=(state:RootState):CategoriesStatusTypes.StatusState=>{ return
state.categoriesStatus }
```

```
src\app\store\categories\categories.types.ts
import { CategoriesStatusTypes as
StatusTypes } from './status';import { Category } from 'src/types/Category';import { NotSyncStateItemBase,SyncStateItemBase } from '../store.types';import { Color } from 'src/types/Color';export type CategoryItemToWithColor<T
extends { color:string }>=Omit<T,'color'> & { color:Color };export type
NotSyncStateItem=NotSyncStateItemBase & Category;export type
NotSyncState=NotSyncStateItem[];export type
CategoryNotSyncStateItemWithColor=CategoryItemToWithColor<NotSyncStateItem>
export type SyncStateItem=SyncStateItemBase & Category;export type
SyncState=SyncStateItem[];export type
CategorySyncStateItemWithColor=CategoryItemToWithColor<SyncStateItem>export
type CategoryStateItem=SyncStateItem|NotSyncStateItem;export type
CategoryStateItemWithColor=[CategoryNotSyncStateItemWithColor|CategorySyncStateItemWithColor];export { StatusTypes }
```

```
src\app\store\categories\index.ts
export { CategoriesActions } from './categories.actions';export { selectCategories } from './categories.select';export * as CategoriesTypes from "
```

```
src\app\store\categories\helpers\category-state-item-with-color-to-default.helper.ts
import { CategoryStateItemWithColor,CategoryStateItem } from '../categories.types';export
function
```



```
categoryStateItemWithColorToDefault(state:CategoryStateItemWithColor):CategoryStateItem{return{...state,color:state.color._id}}
```

```
src\app\store\categories\helpers\change-category-color-id-to-color-object.helper.ts
import{Color}from'src/types/Color';import{CategoryStateItem,CategoryStateItemWithColor}from'../categories.types';export function
changeCategoryColorIdToColorObject(categories:CategoryStateItem[],colors:Color[]):CategoryStateItemWithColor[]{if(colors.length===0)return[];return
categories.map(item=>{const
color=colors.find(colorItem=>colorItem._id===item.color);if(color===undefined){throw new Error('Not found color id in colors')};return{...item,color:color}})as
CategoryStateItemWithColor[]}
```

```
src\app\store\categories\helpers\index.ts
export{changeCategoryColorIdToColorObject}from"
```

```
src\app\store\categories\not-sync\categories-not-sync.actions.ts
import{createActionGroup,props}from '@ngrx/store';import{CategoriesBasicSet}from'src/types/ApiInputs';import{NotSyncStatus}from'../store.types';import{NotSyncState,NotSyncStateItem}from'../categories.types';export const
CategoriesNotSyncActions=createActionGroup({source:'Categories not
sync',events:{set:props<{categories:NotSyncState}>(),add:props<NotSyncStateItem>(),changeStatus:props<{status:NotSyncStatus;category:NotSyncStateItem}>(),delete:props<NotSyncStateItem>(),update:props<{oldCategory:NotSyncStateItem;dataForUpdate:CategoriesBasicSet}>()}}})
```

```
src\app\store\categories\not-sync\categories-not-sync.reducer.ts
import{createReducer,on}from '@ngrx/store';import{NotSyncState}from'../categories.types';import{CategoriesNotSyncActions}from'./categories-not-sync.actions';export const
initialState:NotSyncState=[];export const
categoriesNotSyncReducer=createReducer(initialState,on(CategoriesNotSyncActions.add,(state,payload)=>{return[...state,payload]}),on(CategoriesNotSyncActions.changeStatus,(state,{status,category})=>{return[...state.map(item=>{if(item._id===category._id){return{...item,status:status}};return
item}]]}),on(CategoriesNotSyncActions.delete,(state,payload)=>{return[...state.filter(item=>item._id!==payload._id)]}))
```

```
src\app\store\categories\not-sync\index.ts
export*as NotSyncHelpers
from'./helpers';export{categoriesNotSyncReducer}from'./categories-not-sync.reducer';export{CategoriesNotSyncActions}from"
```

```
src\app\store\categories\not-sync\helpers\change-category-value-to-store-item.helper.ts
```

```
import { NotSyncStatus, NotSyncAction } from 'src/app/store/store.types'; import { CategoriesBasicSet } from 'src/types/ApiInputs'; import { NotSyncStateItem, SyncStateItem } from '.././categories.types'; import { generateNotSyncCategoryId } from './generate-not-sync-category-id.helper'; export function changeAddCategoryValueToStoreItem(data: CategoriesBasicSet): NotSyncStateItem { const forAdd = { ...data, _id: generateNotSyncCategoryId(), status: NotSyncStatus.NOT_SYNCHRONIZED, action: NotSyncAction.CREATED, order: 999 }; return forAdd }; export function changeDeleteCategoryValueToStoreItem(data: SyncStateItem): NotSyncStateItem { return { ...data, status: NotSyncStatus.NOT_SYNCHRONIZED, action: NotSyncAction.DELETED } }; export function changeUpdateCategoryValueToStoreItem(oldCategoryData: SyncStateItem, newCategoryData: Partial<CategoriesBasicSet>): NotSyncStateItem { return { ...oldCategoryData, status: NotSyncStatus.NOT_SYNCHRONIZED, action: NotSyncAction.CHANGED, ...newCategoryData } }
```

```
src\app\store\categories\not-sync\helpers\generate-not-sync-category-id.helper.ts
import { NOT_SYNC_ID_TAG } from '.././categories.config'; export const generateNotSyncCategoryId = () => { return `${NOT_SYNC_ID_TAG}-${new Date(Date.now()).getTime()}` }
```

```
src\app\store\categories\not-sync\helpers\index.ts
export * from "
```

```
src\app\store\categories\status\categories-status.actions.ts
import { createActionGroup, props } from '@ngrx/store'; import { CategoriesStatusTypes } from './'; export const CategoriesStatusActions = createActionGroup({ source: 'Categories status', events: { set: props<{ status: CategoriesStatusTypes.StatusState }>() } })
```

```
src\app\store\categories\status\categories-status.reducer.ts
import { createReducer, on } from '@ngrx/store'; import { CategoriesStatusActions } from './categories-status.actions'; import { CategoriesStatusTypes } from './'; export const initialState: CategoriesStatusTypes.StatusState = CategoriesStatusTypes.StatusState.NOT_SYNCHRONIZED as CategoriesStatusTypes.StatusState; export const categoriesStatusReducer = createReducer(initialState, on(CategoriesStatusActions.set, (state, payload) => { return payload.status })))
```

```
src\app\store\categories\status\categories-status.types.ts
import { LoadStatus } from '.././store.types'; export { LoadStatus as StatusState }
```

```
src\app\store\categories\status\index.ts
```

```
export*as CategoriesStatusTypes from'./categories-
status.types';export{ CategoriesStatusActions }from'./categories-
status.actions';export{ categoriesStatusReducer }from"
```

```
src\app\store\categories\sync\categories-sync.actions.ts
import{ props,createActionGroup }from'@ngrx/store';import{ ReorderCategoryData }fro
m'src/types/ApiInputs';import{ SyncState,SyncStateItem }from'../categories.types';export
const CategoriesSyncActions=createActionGroup({ source:'Categories
sync',events:{ set:props<{ categories:SyncState }>(),add:props<{ category:SyncStateItem
}>(),delete:props<SyncStateItem>(),update:props<{ category:SyncStateItem }>(),orderU
pdate:props<{ data:ReorderCategoryData }>()}})
```

```
src\app\store\categories\sync\categories-sync.reducer.ts
import{ createReducer,on }from'@ngrx/store';import{ CategoriesSyncActions }from'./cate
gories-sync.actions';import{ SyncState }from'../categories.types';export const
initialState:SyncState=[];export const
categoriesReducer=createReducer(initialState,on(CategoriesSyncActions.set,(state,{ cate
gories })=>[...categories]),on(CategoriesSyncActions.add,(state,{ category })=>[...state,c
ategory]),on(CategoriesSyncActions.delete,(state,category)=>{ return[...state.filter(item
=>item._id!==category._id)]}),on(CategoriesSyncActions.orderupdate,(state,{ data })=>
{ return[...state.map(item=>{ if(item._id===data.categoryId){ return{ ...item,order:data.cu
rrentIndex } };return item }]))))
```

```
src\app\store\categories\sync\index.ts
export{ categoriesReducer }from'./categories-
sync.reducer';export{ CategoriesSyncActions }from"
```

```
src\app\store\colors\colors.actions.ts
import{ createAction,props }from'@ngrx/store';import{ Color }from'src/types/Color';cons
t TAG='[API Colors]';export const loadColors=createAction(` ${TAG}Load
Colors`,`(props:{ force:boolean}={ force:false })=>props`);export const
colorsLoadedSuccess=createAction(` ${TAG}Colors Loaded
Success`,`props<{ payload:Color[] }>()`);export const
colorsLoadedError=createAction(` ${TAG}Colors Loaded Error`)
```

```
src\app\store\colors\colors.effects.ts
import{ Injectable }from'@angular/core';import{ Actions,createEffect,ofType }from'@ngr
x/effects';import{ select,Store }from'@ngrx/store';import{ EMPTY }from'rxjs';import{ ma
p,exhaustMap,catchError,withLatestFrom }from'rxjs/operators';import{ ApiService }from
'src/app/services/api.service';import{ ColorsActions }from'.';import{ RootState }from'../ro
otTypes'@Injectable();export class
ColorEffects{ loadColors$=createEffect(()=>this.actions$.pipe(ofType(ColorsActions.lo
adColors),withLatestFrom(this.store.pipe(select('colors'))),exhaustMap(([params,colors
Value])=>{ if(colorsValue.length===0||params.force){ return
```

```
this.api.getAllColors().pipe(map(colors=>ColorsActions.colorsLoadedSuccess({payload:colors})),catchError(()=>EMPTY));return EMPTY}));constructor(private actions$:Actions,private api:ApiService,private store:Store<RootState>){}}
```

```
src\app\store\colors\colors.reducer.ts
import { createReducer, on } from '@ngrx/store'; import { ColorState } from '.'; import { colorsLoadedSuccess } from './colors.actions'; export const initialState: ColorState = []; export const colorsReducer = createReducer(initialState, on(colorsLoadedSuccess, (state, { payload }) => [...payload]))
```

```
src\app\store\colors\colors.select.ts
import { RootState } from '../rootTypes'; export const selectColors = (state: RootState) => state.colors
```

```
src\app\store\colors\colors.types.ts
import { Color } from 'src/types/Color'; export type ColorState = Color[]
```

```
src\app\store\colors\index.ts
export { colorsReducer } from './colors.reducer'; export * as ColorsActions from './colors.actions'; export { ColorState } from './colors.types'; export { selectColors } from "
```

```
src\app\store\statistic\index.ts
export { StatisticActions } from './statistic.actions'; export { selectStatistic } from './statistic.select'; export * as StatisticTypes from "
```

```
src\app\store\statistic\statistic.actions.ts
import { createActionGroup, props } from '@ngrx/store'; import { AddStatisticInputs } from 'src/types/ApiInputs'; import { NotSyncStateItem } from './statistic.types'; import { StatisticStateItemWithCategory } from './statistic.types'; export const StatisticActions = createActionGroup({ source: 'Statistic', events: { load: (props: { force: boolean }) => { force: false } } => props, add: (statistic: AddStatisticInputs) => statistic, delete: (statistic: StatisticStateItemWithCategory) => statistic, update: props < { oldStatistic: StatisticStateItemWithCategory; dataForUpdate: AddStatisticInputs } > (), addEffect: (statistic: NotSyncStateItem) => statistic, deleteEffect: (statistic: NotSyncStateItem) => statistic, updateEffect: (statistic: NotSyncStateItem) => statistic } })
```

```
src\app\store\statistic\statistic.config.ts
export const NOT_SYNC_ID_TAG = "
```

```
src\app\store\statistic\statistic.effects.ts
import { Injectable } from '@angular/core'; import { Actions, createEffect, ofType } from '@ngrx/effects'; import { select, Store } from '@ngrx/store'; import { EMPTY, of } from 'rxjs'; import { map, exhaustMap, catchError, withLatestFrom, mergeMap, switchMap } from 'rxjs/operators'; import { ApiService } from 'src/app/services/api.service'; import { StatisticActions } from '.'; i
```

```

import { StatisticNotSyncActions } from './not-sync/statistic-not-sync.actions';
import { StatisticSyncActions } from './sync/statistic-sync.actions';
import { RootState } from '../rootTypes';
import { NotSyncHelpers } from './not-sync';
import { StatisticStatusActions, StatisticStatusTypes } from './status';
import { NotSyncStateItem } from './statistic.types';
import { NotSyncStatus } from './store.types';
import { statisticStateItemWithCategoryToDefault } from './helpers/statistic-state-item-with-category-to-default.helper';
@Injectable();
export class StatisticEffects {
  loadStatistic$ = createEffect(() => this.actions$.pipe(
    ofType(StatisticActions.load),
    withLatestFrom(this.store.pipe(select('statistic'))),
    exhaustMap(([params, statisticValue]) => {
      if (statisticValue.length === 0 || !params.force) {
        this.store.dispatch(StatisticStatusActions.set({ status: StatisticStatusTypes.StatusState.SYNCHRONIZATION }));
        return this.api.getAllStatisticRecords().pipe(
          mergeMap(value => {
            return [StatisticStatusActions.set({ status: StatisticStatusTypes.StatusState.SYNCHRONIZED }), StatisticSyncActions.set({ statistic: value })];
          }),
          catchError(() => {
            this.store.dispatch(StatisticStatusActions.set({ status: StatisticStatusTypes.StatusState.ERROR }));
            return EMPTY;
          })
        );
      }
      return EMPTY;
    })
  );

  add$ = createEffect(() => this.actions$.pipe(
    ofType(StatisticActions.add),
    exhaustMap(statisticForAdd => {
      const statisticAsNotSyncStateItem: NotSyncStateItem = NotSyncHelpers.changeAddStatisticValueToStoreItem(statisticForAdd);
      return of(StatisticNotSyncActions.add(statisticAsNotSyncStateItem), StatisticActions.addeffect(statisticAsNotSyncStateItem));
    })
  );

  update$ = createEffect(() => this.actions$.pipe(
    ofType(StatisticActions.update),
    exhaustMap(categoryForUpdate => {
      const oldStatistic = statisticStateItemWithCategoryToDefault(categoryForUpdate.oldStatistic);
      if (oldStatistic.status) {
        return [StatisticNotSyncActions.update({ oldStatistic: oldStatistic, dataForUpdate: categoryForUpdate.dataForUpdate })];
      }
      const oldStatisticAsNotSyncStateItem: NotSyncStateItem = NotSyncHelpers.changeUpdateStatisticValueToStoreItem(oldStatistic, categoryForUpdate.dataForUpdate);
      return of(StatisticNotSyncActions.add(oldStatisticAsNotSyncStateItem), StatisticSyncActions.delete(oldStatistic), StatisticActions.updateeffect(oldStatisticAsNotSyncStateItem));
    })
  );

  delete$ = createEffect(() => this.actions$.pipe(
    ofType(StatisticActions.delete),
    exhaustMap(statisticForDeleteInput => {
      const statisticForDelete = statisticStateItemWithCategoryToDefault(statisticForDeleteInput);
      if (statisticForDelete.status) {
        return [StatisticNotSyncActions.delete(statisticForDelete)];
      }
      const statisticAsNotSyncStateItem: NotSyncStateItem = NotSyncHelpers.changeDeleteStatisticValueToStoreItem(statisticForDelete);
      return of(StatisticNotSyncActions.add(statisticAsNotSyncStateItem), StatisticSyncActions.delete(statisticForDelete), StatisticActions.deleteeffect(statisticAsNotSyncStateItem));
    })
  );

  addStatistic$ = createEffect(() => this.actions$.pipe(
    ofType(StatisticActions.addeffect),
    mergeMap(inputStatistic => {
      this.store.dispatch(StatisticNotSyncActions.changestatus({ status: NotSyncStatus.SYNCHRONIZATION, statistic: inputStatistic }));
      this.store.dispatch(StatisticStatusActions.set({ status: StatisticStatusTypes.StatusState.SYNCHRONIZATION }));
    })
  );

  return this.api.addStatisticRecord(inputStatistic).pipe(
    switchMap(resultStatistic => [StatisticSync

```

```

cActions.add({ statistic:resultStatistic } ),StatisticStatusActions.set({ status:StatisticStatus
Types.StatusState.SYNCHRONIZED } ),StatisticNotSyncActions.delete(inputStatistic))]
,catchError(()=>{ this.store.dispatch(StatisticNotSyncActions.changestatus({ status:NotS
yncStatus.ERROR,statistic:inputStatistic } ));this.store.dispatch(StatisticStatusActions.se
t({ status:StatisticStatusTypes.StatusState.ERROR } ));return
EMPTY } ))))));deleteStatistic$=createEffect(()=>this.actions$.pipe(ofType(StatisticActi
ons.deleteeffect),mergeMap(inputStatistic=>{ this.store.dispatch(StatisticNotSyncActio
ns.changestatus({ status:NotSyncStatus.SYNCHRONIZATION,statistic:inputStatistic } ))
;this.store.dispatch(StatisticStatusActions.set({ status:StatisticStatusTypes.StatusState.S
YNCHRONIZATION } ));return
this.api.deleteStatistic(inputStatistic._id).pipe(switchMap(()=>[StatisticStatusActions.se
t({ status:StatisticStatusTypes.StatusState.SYNCHRONIZED } ),StatisticNotSyncActions
.delete(inputStatistic)]),catchError(()=>{ this.store.dispatch(StatisticNotSyncActions.cha
ngestatus({ status:NotSyncStatus.ERROR,statistic:inputStatistic } ));this.store.dispatch(St
atisticStatusActions.set({ status:StatisticStatusTypes.StatusState.ERROR } ));return
EMPTY } ))))));updateStatistic$=createEffect(()=>this.actions$.pipe(ofType(StatisticActi
ons.updateeffect),mergeMap(inputStatistic=>{ this.store.dispatch(StatisticNotSyncActi
ons.changestatus({ status:NotSyncStatus.SYNCHRONIZATION,statistic:inputStatistic }
));this.store.dispatch(StatisticStatusActions.set({ status:StatisticStatusTypes.StatusState.
SYNCHRONIZATION } ));return
this.api.updateStatistic(inputStatistic._id,inputStatistic).pipe(switchMap(resultStatistic=
>[StatisticSyncActions.add({ statistic:resultStatistic } ),StatisticStatusActions.set({ status:
StatisticStatusTypes.StatusState.SYNCHRONIZED } ),StatisticNotSyncActions.delete(i
nputStatistic)]),catchError(()=>{ this.store.dispatch(StatisticNotSyncActions.changestat
us({ status:NotSyncStatus.ERROR,statistic:inputStatistic } ));this.store.dispatch(StatisticS
tatusActions.set({ status:StatisticStatusTypes.StatusState.ERROR } ));return
EMPTY } ))))));constructor(private actions$:Actions,private api:ApiService,private
store:Store<RootState>){ }

```

src\app\store\statistic\statistic.select.ts

```

import { RootState } from '../rootTypes';import { NotSyncState,StatisticStateItemWithCateg
ory,SyncState } from './statistic.types';import { StatisticStatusTypes } from './status';import { c
hangeStatisticCategoryIdToCategoryObject } from './helpers/change-statistic-category-id-
to-category-
object.helper';import { createSelector } from '@ngrx/store';import { selectCategories } from 's
rc/app/store/categories/categories.select';import { CategoryStateItemWithColor } from 'src/
app/store/categories/categories.types';import { StatisticStateItem,StatisticNotSyncStateIte
mWithCategory,NotSyncStateItem } from './statistic.types';const
selectNotSyncStatistic=(state:RootState)=>{ return state.notSyncStatistic };const
selectSyncStatistic=(state:RootState)=>{ return state.statistic };export const
selectNotSyncStatisticWithCategory=createSelector(selectNotSyncStatistic,selectCateg
ories,(notSyncStatistic:NotSyncState,categories:CategoryStateItemWithColor[]):Statisti
cNotSyncStateItemWithCategory[]=>{ return
changeStatisticCategoryIdToCategoryObject<NotSyncStateItem>(notSyncStatistic,cate

```

```

gories)as StatisticNotSyncStateItemWithCategory[]]);export const
selectStatistic=createSelector(selectSyncStatistic,selectNotSyncStatistic,selectCategorie
s,(syncStatistic:SyncState,notSyncStatistic:NotSyncState,categories:CategoryStateItem
WithColor[]):StatisticStateItemWithCategory[]=>{ return
changeStatisticCategoryIdToCategoryObject([...syncStatistic,...notSyncStatistic],catego
ries)});export const
selectStatisticStatus=(state:RootState):StatisticStatusTypes.StatusState=>{ return
state.statisticStatus }

```

```

src\app\store\statistic\statistic.types.ts
import { StatisticStatusTypes as
StatusTypes } from './status';import { Statistic } from 'src/types/Statistic';import { NotSyncSta
teItemBase,SyncStateItemBase } from './store.types';import { CategoryStateItemWithColo
r } from '../categories/categories.types';export type StatisticItemToWithCategory<T
extends { category:string }>=Omit<T,'category'>& { category:CategoryStateItemWithColo
r};export type NotSyncStateItem=NotSyncStateItemBase&Statistic;export type
NotSyncState=NotSyncStateItem[];export type
StatisticNotSyncStateItemWithCategory=StatisticItemToWithCategory<NotSyncStateIt
em>export type SyncStateItem=SyncStateItemBase&Statistic;export type
SyncState=SyncStateItem[];export type
StatisticSyncStateItemWithCategory=StatisticItemToWithCategory<SyncStateItem>ex
port type StatisticStateItem=NotSyncStateItem|SyncStateItem;export type
StatisticStateItemWithCategory=|StatisticNotSyncStateItemWithCategory|StatisticSync
StateItemWithCategory;export { StatusTypes }

```

```

src\app\store\statistic\helpers\change-statistic-category-id-to-category-object.helper.ts
import { CategoryStateItemWithColor } from 'src/app/store/categories/categories.types';im
port { StatisticItemToWithCategory,StatisticStateItem } from '../statistic.types';export
function changeStatisticCategoryIdToCategoryObject<T extends
StatisticStateItem,Y=StatisticItemToWithCategory<T>>(data:StatisticStateItem[],categ
ories:CategoryStateItemWithColor[]):Y[] { if(categories.length===0)return[];return
data.map(item=>{ const
category=categories.find(categoryItem=>categoryItem._id===item.category);if(categor
y===undefined){ return null };return { ...item,category:category,date:new
Date(item.date).toISOString() } as Y }).filter(item=>item!==null)as Y[] }

```

```

src\app\store\statistic\helpers\index.ts
export { changeStatisticCategoryIdToCategoryObject } from './change-statistic-category-
id-to-category-object.helper';export { statisticStateItemWithCategoryToDefault } from "

```

```

src\app\store\statistic\helpers\statistic-state-item-with-category-to-default.helper.ts
import { StatisticStateItemWithCategory,StatisticStateItem } from '../statistic.types';export
function

```

```

statisticStateItemWithCategoryToDefault(state:StatisticStateItemWithCategory):StatisticStateItem{
  return{...state,category:state.category._id}}

src\app\store\statistic\not-sync\index.ts
export{ statisticNotSyncReducer}from'./statistic-not-sync.reducer';export{ StatisticNotSyncActions }from'./statistic-not-sync.actions';export*as NotSyncHelpers from"

src\app\store\statistic\not-sync\statistic-not-sync.actions.ts
import{ createActionGroup,props }from'@ngrx/store';import{ AddStatisticInputs }from'src/types/ApiInputs';import{ NotSyncStatus }from'../../store.types';import{ NotSyncState,NotSyncStateItem }from'./statistic.types';export const
StatisticNotSyncActions=createActionGroup({ source:'Statistic not sync',events:{ set:props<{ statistic:NotSyncState }>(),add:props<NotSyncStateItem>(),changeStatus:props<{ status:NotSyncStatus;statistic:NotSyncStateItem }>(),delete:props<NotSyncStateItem>(),update:props<{ oldStatistic:NotSyncStateItem;dataForUpdate:AddStatisticInputs }>()}})

src\app\store\statistic\not-sync\statistic-not-sync.reducer.ts
import{ createReducer,on }from'@ngrx/store';import{ NotSyncState }from'./statistic.types';import{ StatisticNotSyncActions }from'./statistic-not-sync.actions';export const
initialState:NotSyncState=[];export const
statisticNotSyncReducer=createReducer(initialState,on(StatisticNotSyncActions.add,(state,payload)=>{ return[...state,payload]}),on(StatisticNotSyncActions.changeStatus,(state,{ status,statistic })=>{ return[...state.map(item=>{ if(item._id===statistic._id){ return{ ...item,status:status } };return item}]}),on(StatisticNotSyncActions.delete,(state,payload)=>{ return[...state.filter(item=>item._id!==payload._id)]}))

src\app\store\statistic\not-sync\helpers\change-statistic-value-to-store-item.helper.ts
import{ generateNotSyncStatisticId }from'./generate-not-sync-statistic-id.helper';import{ AddStatisticInputs }from'src/types/ApiInputs';import{ NotSyncStateItem,SyncStateItem }from'../../statistic.types';import{ NotSyncStatus,NotSyncAction }from'src/app/store/store.types';export function
changeAddStatisticValueToStoreItem(data:AddStatisticInputs):NotSyncStateItem{ const
forAdd={ ...data,_id:generateNotSyncStatisticId(),status:NotSyncStatus.NOT_SYNCHRONIZED,action:NotSyncAction.CREATED};return forAdd};export function
changeDeleteStatisticValueToStoreItem(data:SyncStateItem):NotSyncStateItem{ return
{ ...data,date:new
Date(data.date).toISOString(),status:NotSyncStatus.NOT_SYNCHRONIZED,action:NotSyncAction.DELETED}};export function
changeUpdateStatisticValueToStoreItem(data:SyncStateItem,newData:AddStatisticInput

```



```

ts):NotSyncStateItem{return {...data,status:NotSyncStatus.NOT_SYNCHRONIZED,action:NotSyncAction.DELETED,...newData}}

src\app\store\statistic\not-sync\helpers\generate-not-sync-statistic-id.helper.ts
import{NOT_SYNC_ID_TAG}from'../../statistic.config';export const
generateNotSyncStatisticId=()=>{return`${NOT_SYNC_ID_TAG}-${new
Date(Date.now()).getTime()}`}

src\app\store\statistic\not-sync\helpers\index.ts
export*from"

src\app\store\statistic\status\index.ts
export*as StatisticStatusTypes from'./statistic-
status.types';export{StatisticStatusActions}from'./statistic-
status.actions';export{statisticStatusReducer}from"

src\app\store\statistic\status\statistic-status.actions.ts
import{createActionGroup,props}from'@ngrx/store';import{StatisticStatusTypes}from'.
';export const StatisticStatusActions=createActionGroup({source:'Statistic
status',events:{set:props<{status:StatisticStatusTypes.StatusState}>()}})

src\app\store\statistic\status\statistic-status.reducer.ts
import{createReducer,on}from'@ngrx/store';import{StatisticStatusActions}from'./statistic-
status.actions';import{StatisticStatusTypes}from'.';export const
initialState:StatisticStatusTypes.StatusState=StatisticStatusTypes.StatusState.NOT_SY
NCHRONIZED as StatisticStatusTypes.StatusState;export const
statisticStatusReducer=createReducer(initialState,on(StatisticStatusActions.set,(state,pay
load)=>{return payload.status}))

src\app\store\statistic\status\statistic-status.types.ts
import{LoadStatus}from'../../store.types';export{LoadStatus as StatusState}

src\app\store\statistic\sync\index.ts
export{statisticReducer}from'./statistic-
sync.reducer';export{StatisticSyncActions}from"

src\app\store\statistic\sync\statistic-sync.actions.ts
import{props,createActionGroup}from'@ngrx/store';import{SyncState,SyncStateItem}f
rom'../../statistic.types';export const
StatisticSyncActions=createActionGroup({source:'Statistic
sync',events:{set:props<{statistic:SyncState}>(),add:props<{statistic:SyncStateItem}>()
,delete:props<SyncStateItem>(),update:props<{category:SyncStateItem}>()}})

src\app\store\statistic\sync\statistic-sync.reducer.ts

```

```
import { createReducer, on } from '@ngrx/store'; import { StatisticSyncActions } from './statistic-sync.actions'; import { SyncState } from './statistic.types'; export const initialState: SyncState = []; export const statisticReducer = createReducer(initialState, on(StatisticSyncActions.set, (state, { statistic }) => { return [...statistic] }), on(StatisticSyncActions.add, (state, { statistic }) => [...state, statistic]), on(StatisticSyncActions.delete, (state, statistic) => { return [...state.filter(item => item._id !== statistic._id)] })))
```

```
src\environments\environment.development.ts
export const environment = { version: '1.0.0', type: 'dev', googleClientId: '446806438760-0h3fb1c15rjdl05rqtui4ki0lqpqlr08.apps.googleusercontent.com', API_HOST: '}
```

```
src\environments\environment.next.ts
export const environment = { version: '1.1.0', type: 'next', googleClientId: '446806438760-0h3fb1c15rjdl05rqtui4ki0lqpqlr08.apps.googleusercontent.com', API_HOST: '}
```

```
src\environments\environment.ts
export const
environment = { version: '1.0.2', type: 'production', googleClientId: '446806438760-0h3fb1c15rjdl05rqtui4ki0lqpqlr08.apps.googleusercontent.com', API_HOST: '}
```

```
src\types\ApiInputs.ts
import { Category } from './Category'; import { Statistic } from './Statistic'; import { User } from './User'; export type
CategoriesBasicSet = Omit<Category, 'color' | 'order' | '_id'> & { color: string }; export type
AddStatisticInputs = Omit<Statistic, '_id'> export type InitializeSuccess = User; export type
InitializeFailed = { authorized: false }; export interface
ReorderCategoryData { categoryId: string; previousIndex: number; currentIndex: number }; export type
ReorderCategoryReturnData = ReorderCategoryData[]
```

```
src\types\Category.ts
export interface
Category { _id: string; name: string; comment: string; color: string; order: number; dimension?: string }
```

```
src\types\Color.ts
export interface Color { _id: string; name: string; colorHEX: string; order: number }
```

```
src\types\Statistic.ts
export interface
Statistic { _id: string; date: string; count: number; comment: string; category: string; summ: number }
```

```
src\types\User.ts
```

export interface

```
UserBase{ _id:string;email:string;family_name:string;given_name:string;name:string;picture:string};export interface User extends UserBase{ authorized:boolean}
```

Backend:

src\app.controller.spec.ts

```
import { Test, TestingModule } from '@nestjs/testing';import { AppController } from './app.controller';import { AppService } from './app.service';describe('AppController',()=>{let appController:AppController;beforeEach(async()=>{const app:TestingModule=await Test.createTestingModule({ controllers:[AppController], providers:[AppService]}).compile();appController=app.get<AppController>(AppController);describe('root',()=>{it('should return "Hello World!"',()=>{expect(appController.getHello()).toBe('Hello World!')}))})})})
```

src\app.controller.ts

```
import { Controller, Get, Post } from '@nestjs/common';import { AppService } from './app.service';@Controller();export class AppController{ constructor(private readonly appService:AppService){ } @Get();getHello():string{ return this.appService.getHello() } @Post();postHello():string{ return'post hello' } }
```

src\app.interfaces.ts

```
import mongoose from 'mongoose';export type UserIdSession=|mongoose.Schema.Types.ObjectId|mongoose.Types.ObjectId;type SessionDataAuthorized={ authorized:true;userId:UserIdSession };type SessionDataNotAuthorized={ authorized:undefined;userId:undefined };type SessionDataAuth=SessionDataAuthorized|SessionDataNotAuthorized;declare module 'express-session'{ interface SessionData{ auth?:SessionDataAuth } }
```

src\app.module.ts

```
import { Module } from '@nestjs/common';import { AppController } from './app.controller';import { AppService } from './app.service';import { ConfigModule, ConfigService } from '@nestjs/config';import { ColorModule } from './color/color.module';import { MongooseModule } from '@nestjs/mongoose';import { environmentConfig } from './config/environment.config';import { UserModule } from './user/user.module';import { AuthModule } from './auth/auth.module';import { InitializeModule } from './initialize/initialize.module';import { CategoryModule } from './category/category.module';import { StatisticModule } from './statistic/statistic.module';import { CategoryGroupModule } from './category-group/category-group.module';@Module({ imports:[ConfigModule.forRoot({ envFilePath:['.env',process.env.NODE_ENV?.env.local:'.env.development.local'],load:[environmentConfig]}),MongooseModule.forRootAsync({ imports:[ConfigModule],inject:[ConfigService],useFactory:async(config:ConfigService)=>({ uri:config.get<string>('database.uri') })),ColorModule,UserModule,AuthModule,InitializeModule,CategoryModule,StatisticModule,CategoryGroupModule], controllers:[AppController], providers:[AppService]});export class AppModule{ }
```

```
src\app.service.ts
import { Injectable } from '@nestjs/common';
export class AppService {
  getHello(): string {
    return `Hello World! Current port ${process.env.PORT}`;
  }
}
```

```
src/main.ts
import { NestFactory } from '@nestjs/core';
import { AppModule } from './app.module';
import { SwaggerModule, DocumentBuilder } from '@nestjs/swagger';
import { getHttpsOptions } from './helpers/getHttpsOptions.helper';
import { COOKIE_MAX_AGE } from './config/constants.config';
import session from 'express-session';
import MongoStore from 'connect-mongo';
async function bootstrap() {
  const app = await NestFactory.create(AppModule, {
    httpsOptions: process.env.NODE_ENV ? undefined : getHttpsOptions()
  });
  app.enableCors({
    origin: ['http:', 'https:'],
    methods: ['GET', 'POST', 'PUT', 'DELETE', 'OPTIONS'],
    credentials: true
  });
  const config = new DocumentBuilder()
    .setTitle('Counter')
    .setDescription('The Counter API description')
    .setVersion('1.0.0')
    .build();
  const document = SwaggerModule.createDocument(app, config);
  SwaggerModule.setup('api', app, document);
  const store = MongoStore.create({
    mongoUrl: process.env.MONGODB_URI
  });
  const sessionConfig: session.SessionOptions = {
    proxy: true,
    secret: 'keyboard cat',
    name: 'sessionId',
    resave: false,
    saveUninitialized: false,
    cookie: {
      maxAge: COOKIE_MAX_AGE,
      secure: process.env.NODE_ENV === 'production' || process.env.NODE_ENV === 'production' || process.env.NODE_ENV === 'production'
    },
    httpOnly: true,
    sameSite: 'none'
  };
}
```

```
src/auth/auth.controller.ts
import { Controller, UseGuards, Req, Res, HttpStatus, Post, BadRequestException, Session } from '@nestjs/common';
import { Request, Response } from 'express';
import { AuthService } from './auth.service';
import { JwtPayloadUser } from './auth.interface';
import { JwtAuthGuard } from './guard/jwt-auth.guard';
import { SessionData } from 'express-session';
@Controller('auth')
export class AuthController {
  constructor(private authService: AuthService) {}
  @Post()
  @UseGuards(JwtAuthGuard)
  async auth(@Req() req: Request, @Session() session: SessionData, @Res() res: Response) {
    if (!req.user) {
      return new BadRequestException('No authorization header with jwt-token');
    }
    const userDocument = await this.authService.signIn(session, req.user as JwtPayloadUser);
    res.status(HttpStatus.OK).json(userDocument);
  }
  @Post('logout')
  async logout(@Req() req: Request, @Res() res: Response) {
    req.session.destroy(err => {
      console.log('destroy user session error:', err);
    });
    res.status(HttpStatus.OK).json({ status: 'ok' });
  }
}
```

src\auth\auth.interface.ts

export type

JwtPayloadBasicUser={ iss:string; nbf:number; aud:string; sub:string; email:string; email_verified:boolean; azp:string; name:string; picture:string; given_name:string; family_name:string; iat:number; exp:number; jti:string }; export type

JwtPayloadUser=Pick<JwtPayloadBasicUser,'email'|'email_verified'|'name'|'family_name'|'given_name'|'picture'>

src\auth\auth.module.ts

```
import { Module } from '@nestjs/common'; import { JwtModule } from '@nestjs/jwt'; import { PassportModule } from '@nestjs/passport'; import { AuthController } from './auth.controller'; import { AuthService } from './auth.service'; import { JwtStrategy } from './strategies/jwt.strategy'; import { UserModule } from '../user/user.module'; import { ConfigModule } from '@nestjs/config'; @Module({ imports: [ConfigModule, PassportModule, JwtModule, UserModule], controllers: [AuthController], providers: [AuthService, JwtStrategy] }); export class AuthModule { }
```

src\auth\auth.service.ts

```
import { Injectable, Req, Session } from '@nestjs/common'; import { JwtPayloadUser } from './auth.interface'; import { UserService } from '../user/user.service'; import { SessionData } from 'express-session'; @Injectable(); export class AuthService { constructor(private userService: UserService) { }; async signIn(@Session() session: SessionData, user: JwtPayloadUser): Promise<any> { const userDocument = await this.userService.findOneOrCreate(user); session.auth = { authorized: true, userId: userDocument._id }; return { authorized: true, ...userDocument } } }
```

src\auth\guard\jwt-auth.guard.ts

```
import { Injectable } from '@nestjs/common'; import { AuthGuard } from '@nestjs/passport'; @Injectable(); export class JwtAuthGuard extends AuthGuard('jwt') { }
```

src\auth\strategies\jwt.strategy.ts

```
import { ExtractJwt, Strategy } from 'passport-jwt'; import { PassportStrategy } from '@nestjs/passport'; import { Injectable } from '@nestjs/common'; import { passportJwtSecret } from 'jwks-rsa'; import { JwtPayloadBasicUser, JwtPayloadUser } from '../auth.interface'; import { ConfigService } from '@nestjs/config'; @Injectable(); export class JwtStrategy extends PassportStrategy(Strategy, 'jwt') { constructor(private configService: ConfigService) { super({ secretOrKeyProvider: passportJwtSecret({ cache: true, rateLimit: true, jwksRequestsPerMinute: 5, jwksUri: configService.get<string>('GOOGLE_JWK_PUBLIC_KEYS') || '' }), jwtFromRequest: ExtractJwt.fromHeader('authorization'), algorithms: ['RS256'], ignoreExpiration: false } }); async validate(payload: JwtPayloadBasicUser): Promise<JwtPayloadUser> { return { email: payload } }
```

```
ad.email,email_verified:payload.email_verified,name:payload.name,picture:payload.picture,given_name:payload.given_name,family_name:payload.family_name}}}
```

```
src/category/category.config.ts
import { ColorsNames } from 'src/color/color.interface'; export const
DEFAULT_CATEGORIES=[{ name:'Pull-ups',comment:'Pull ups
count',color:ColorsNames.RED,order:1},{ name:'Push-ups',comment:'Push ups
count',color:ColorsNames.EMERALD,order:2},{ name:'Wall slides',comment:'Pull ups
count',color:ColorsNames.BLUE,order:3}];export const PROJECTIONS=`__v-
createdAt-updatedAt`
```

```
src/category/category.controller.ts
import { Body, Controller, Delete, Get, HttpStatus, Param, Post, Put, Res, Session } from '@nestjs/common'; import { CategoryService } from './category.service'; import { SessionData } from 'express-session'; import { Response } from 'express'; import { CreateCategoryDto } from './dto/create-category.dto'; import { UpdateCategoryDto } from './dto/update-category.dto'; import { ReorderCategoryDto } from './dto/reorder-category.dto'; @Controller('category'); export class CategoryController { constructor(private categoryService: CategoryService) {} @Get('all'); async getAll(@Session() session: SessionData, @Res() res: Response) { if (session.auth === undefined || !session.auth.authorized) { res.status(HttpStatus.UNAUTHORIZED).json({ message: 'Unauthorized' }); return }; res.status(HttpStatus.OK).json(await this.categoryService.getAll(session.auth.userId)) } @Post('add'); async addNew(@Session() session: SessionData, @Res() res: Response, @Body() body: CreateCategoryDto) { if (session.auth === undefined || !session.auth.authorized) { res.status(HttpStatus.UNAUTHORIZED).json({ message: 'Unauthorized' }); return }; res.status(HttpStatus.OK).json(await this.categoryService.add(body, session.auth.userId)) } @Delete(':id'); async delete(@Param('id') id: string, @Session() session: SessionData, @Res() res: Response) { if (session.auth === undefined || !session.auth.authorized) { res.status(HttpStatus.UNAUTHORIZED).json({ message: 'Unauthorized' }); return }; res.status(HttpStatus.OK).json(await this.categoryService.delete(id, session.auth.userId)) } @Put('reorder'); async reorder(@Body() body: ReorderCategoryDto, @Session() session: SessionData, @Res() res: Response) { if (session.auth === undefined || !session.auth.authorized) { res.status(HttpStatus.UNAUTHORIZED).json({ message: 'Unauthorized' }); return }; const result = await this.categoryService.reorder(body, session.auth.userId); if (result === null) { return res.status(HttpStatus.BAD_REQUEST).json({ message: 'Something gonna wrong' }); }; res.status(HttpStatus.OK).json(result) } @Put(':id'); async put(@Param('id') id: string, @Body() body: UpdateCategoryDto, @Session() session: SessionData, @Res() res: Response) { if (session.auth === undefined || !session.auth.authorized) { res.status(HttpStatus.UNAUTHORIZED).json({ message: 'Unauthorized' }); return }; const result = await this.categoryService.edit(id, body, session.auth.userId); if (result === null) { return
```

```
res.status(HttpStatus.BAD_REQUEST).json({ message: 'Something gonna wrong' }));res.status(HttpStatus.OK).json(result)}}
```

```
src\category\category.interface.ts
```

```
import { HydratedDocument } from 'mongoose';import { Category } from './category.schema';export type ICategory = HydratedDocument<Category>
```

```
src\category\category.module.ts
```

```
import { Module } from '@nestjs/common';import { CategoryController } from './category.controller';import { CategoryService } from './category.service';import { MongooseModule } from '@nestjs/mongoose';import { Category, CategorySchema } from './category.schema';import { ColorModule } from 'src/color/color.module'@Module({ imports: [ColorModule, MongooseModule.forFeature([ { name: Category.name, schema: CategorySchema } ])], controllers: [CategoryController], providers: [CategoryService], exports: [CategoryService] });export class CategoryModule { }
```

```
src\category\category.schema.ts
```

```
import { Prop, Schema, SchemaFactory } from '@nestjs/mongoose';import mongoose from 'mongoose';import { Color } from 'src/color/color.schema';import { User } from 'src/user/user.schema';import { Group } from 'src/category-group/category-group.schema'@Schema();export class Category { @Prop({ type: mongoose.Schema.Types.ObjectId, ref: 'User', required: true });user: User @Prop({ required: true });name: string @Prop();comment?: string @Prop({ type: mongoose.Schema.Types.ObjectId, ref: 'Color', required: true });color: Color @Prop({ required: true });order: number @Prop();dimension?: string @Prop({ type: [mongoose.Schema.Types.ObjectId], ref: 'Group' });group?: [Group] };export const CategorySchema = SchemaFactory.createForClass(Category)
```

```
src\category\category.service.ts
```

```
import { Body, Injectable } from '@nestjs/common';import { Category } from './category.schema';import { InjectModel } from '@nestjs/mongoose';import { Model } from 'mongoose';import { ICategory } from './category.interface';import { CreateCategoryDto } from './dto/create-category.dto';import { ColorService } from './color/color.service';import { DEFAULT_CATEGORIES, PROJECTIONS } from './category.config';import { UserIdSession } from 'src/app.interfaces';import { UpdateCategoryDto } from './dto/update-category.dto';import { ReorderCategoryDto } from './dto/reorder-category.dto'@Injectable();export class CategoryService { constructor( @InjectModel(Category.name) private categoryModel: Model<ICategory>, private colorService: ColorService ) { }; async add( @Body() createCategoryDto: CreateCategoryDto, userId: UserIdSession ) { const lastOrderId = await this.getLastOrder(userId);const dataForAdd = { user: userId, name: createCategoryDto.name, color: createCategoryDto.color, comment: createCategoryDto.comment, dimension: createCategoryDto.dimension, order: lastOrderId === null ? 0 : lastOrderId + 1 };const newCategory = await new
```

```

this.categoryModel(dataForAdd);const newCategoryDocument=await
newCategory.save();return newCategoryDocument};async
getAll(userId:UserIdSession):Promise<Category[]>{const
userAllCategories=this.categoryModel.find({user:userId},PROJECTIONS);return await
userAllCategories.lean()};async
delete(id:string,userId:UserIdSession):Promise<unknown>{return
this.categoryModel.deleteOne({_id:id,user:userId})};async
edit(id:string,body:UpdateCategoryDto,userId:UserIdSession){const
updatedCategory=await
this.categoryModel.findByIdAndUpdate({_id:id,user:userId},body,{new:true});if(!upda
tedCategory){return null};const updatedCategoryDocument=await
updatedCategory.save();return updatedCategoryDocument};private async
getLastOrder(userId:UserIdSession){const allUserCategories=await
this.getAll(userId);const
result=allUserCategories.reduce((prev,category)=>(category.order>prev?category.order
:prev),-1);return result===-1?null:result};async
initializeUserDefaultCategories(userId:UserIdSession){const colors=await
this.colorService.initializeDefaultColors();const
defaultCategoriesWithColorId=DEFAULT_CATEGORIES.map(item=>({...item,color:
colors[item.color]}));return
defaultCategoriesWithColorId.map(defaultCategory=>{return
this.add(defaultCategory,userId)});};async
reorder(body:ReorderCategoryDto,userId:UserIdSession):Promise<ReorderCategoryDt
o[]|null>{if(body.currentIndex===body.previousIndex)return null;const
categoriesForUpdateOrder=await
this.getCategoriesByOrderInRange(userId,body.previousIndex,body.currentIndex);cons
t updatedCategoriesForUpdateOrder=await
this.changeCategoriesOrders(categoriesForUpdateOrder,body.previousIndex);const
result:ReorderCategoryDto[]=await
this.transformToReorderResult(updatedCategoriesForUpdateOrder);await
this.updateOrders(result,userId);return result};private async
getCategoriesByOrderInRange(userId:UserIdSession,previousIndex:number,currentInd
ex:number):Promise<ICategory[]>{const
order=previousIndex<currentIndex?{$gte:previousIndex,$lte:currentIndex}:{$gte:curre
ntIndex,$lte:previousIndex};const sort={order:previousIndex<currentIndex?1:-
1};return await
this.categoryModel.find({order:order,user:userId},{},{sort:sort}).lean();private async
changeCategoriesOrders(categories:(ICategory&{previousIndex?:number})[],previousI
ndex:number){const lastIndex=categories.at(-
1)?.order;if(lastIndex===undefined){throw new Error('ERR')};for(let
i=categories.length-2;i>=0;i--){const currentElement=categories.at(i);const
indexNextElement=i+1>categories.length-1?0:i+1;const
nextElement=categories.at(indexNextElement);if(currentElement===undefined||nextEle
ment===undefined){throw new Error(`currentElement||nextElement is undefined in

```



```

reorder.categories=${JSON.stringify(categories)}`));nextElement.previousIndex=nextElement.order;nextElement.order=currentElement.order};categories[0].previousIndex=previousIndex;categories[0].order=lastIndex;return categories};private async
transformToReorderResult(categories:(ICategory&{previousIndex?:number})[]):Promise<ReorderCategoryDto[]>{return
categories.map(item=>{if(!item.previousIndex){throw new Error(`item.previousIndex
is undefined in
transformToReorderResult.categories=${JSON.stringify(categories)}`)};return{category
Id:item._id.toString(),currentIndex:item.order,previousIndex:item.previousIndex}}});private async
updateOrders(categories:ReorderCategoryDto[],userId:UserIdSession){await
Promise.all(categories.map(async item=>{const updatedCategory=await
this.categoryModel.findByIdAndUpdate({_id:item.categoryId,user:userId},{order:item.
currentIndex},{new:true}).lean();if(!updatedCategory){throw new Error(`Error update
categories reorder.Category=${JSON.stringify(item)}`)};return updatedCategory})))}

```

src\category\dto\create-category.dto.ts

```

import{ApiProperty}from'@nestjs/swagger';import{isArray,isEmpty,isString}from'
class-validator';export class
CreateCategoryDto{ @ApiProperty()@IsNotEmpty()@IsString();name:string@ApiProp
erty()@IsString();comment?:string@ApiProperty()@IsNotEmpty()@IsString();color:str
ing@ApiProperty()@IsString();dimension?:string@ApiProperty()@isArray();group?:[s
tring]}

```

src\category\dto\reorder-category.dto.ts

```

import{ApiProperty}from'@nestjs/swagger';import{isEmpty,isString,isNumber}fro
m'class-validator';export class
ReorderCategoryDto{ @ApiProperty()@IsNotEmpty()@IsString();categoryId:string@A
piProperty()@IsNotEmpty()@IsNumber();previousIndex:number@ApiProperty()@IsN
otEmpty()@IsNumber();currentIndex:number}

```

src\category\dto\update-category.dto.ts

```

import{PartialType}from'@nestjs/mapped-
types';import{CreateCategoryDto}from'./create-category.dto';export class
UpdateCategoryDto extends PartialType(CreateCategoryDto){}

```

src\category-group\category-group.config.ts

```

export const PROJECTIONS=`__v-createdAt-updatedAt`

```

src\category-group\category-group.controller.ts

```

import{Body,Controller,Delete,Get,HttpStatus,Param,Post,Put,Res,Session}from'@nest
js/common';import{Response}from'express';import{SessionData}from'express-
session';import{CategoryGroupService}from'./category-
group.service';import{CreateCategoryGroupDto}from'./dto/create-category-

```

```

group.dto';import { UpdateCategoryGroupDto } from './dto/update-category-
group.dto'@Controller('group');export class
CategoryGroupController { constructor(private
categoryGroupService: CategoryGroupService) { } @Get('all');async
all( @Session() session: SessionData, @Res() res: Response) { if(session.auth===undefined||
!session.auth.authorized) { res.status(HttpStatus.UNAUTHORIZED).json({ message: 'Un
authorized' });return };res.status(HttpStatus.OK).json(await
this.categoryGroupService.getAll(session.auth.userId)) } @Post('add');async
add( @Session() session: SessionData, @Res() res: Response, @Body() body: CreateCategor
yGroupDto) { if(session.auth===undefined||!session.auth.authorized) { res.status(HttpStat
us.UNAUTHORIZED).json({ message: 'Unauthorized' });return };res.status(HttpStatus.O
K).json(await
this.categoryGroupService.add(body,session.auth.userId)) } @Put(':id');async
edit( @Session() session: SessionData, @Res() res: Response, @Param('id') id:string, @Body
() body: UpdateCategoryGroupDto) { if(session.auth===undefined||!session.auth.authorize
d) { res.status(HttpStatus.UNAUTHORIZED).json({ message: 'Unauthorized' });return };co
nst result=await
this.categoryGroupService.edit(id,body,session.auth.userId);if(result===null) { return
res.status(HttpStatus.BAD_REQUEST).json({ message: 'Something gonna
wrong' }); };res.status(HttpStatus.OK).json(result) } @Delete(':id');async
delete( @Session() session: SessionData, @Res() res: Response, @Param('id') id:string) { if(s
ession.auth===undefined||!session.auth.authorized) { res.status(HttpStatus.UNAUTHOR
IZED).json({ message: 'Unauthorized' });return };res.status(HttpStatus.OK).json(await
this.categoryGroupService.delete(id,session.auth.userId)) } }

```

```

src\category-group\category-group.interface.ts
import { HydratedDocument } from 'mongoose';import { Group } from './category-
group.schema';export type IGroup=HydratedDocument<Group>

```

```

src\category-group\category-group.module.ts
import { Module } from '@nestjs/common';import { MongooseModule } from '@nestjs/mong
oose';import { CategoryGroupService } from './category-
group.service';import { CategoryGroupController } from './category-
group.controller';import { Group,GroupSchema } from './category-
group.schema'@Module({ imports:[MongooseModule.forFeature([ { name:Group.name,s
chema:GroupSchema } ])],providers:[CategoryGroupService],controllers:[CategoryGrou
pController],exports:[CategoryGroupService] });export class CategoryGroupModule { }

```

```

src\category-group\category-group.schema.ts
import { Prop,Schema,SchemaFactory } from '@nestjs/mongoose'@Schema();export class
Group { @Prop({ required:true });name:string@Prop({ required:true });order:number };exp
ort const GroupSchema=SchemaFactory.createForClass(Group)

```

```

src\category-group\category-group.service.ts

```

```

import { Body, Injectable } from '@nestjs/common'; import { InjectModel } from '@nestjs/mongoose'; import { Model } from 'mongoose'; import { UserIdSession } from 'src/app/interfaces'; import { IGroup } from './category-group.interface'; import { Group } from './category-group.schema'; import { PROJECTIONS } from './category-group.config'; import { CreateCategoryGroupDto } from './dto/create-category-group.dto'; import { UpdateCategoryGroupDto } from './dto/update-category-group.dto'; @Injectable(); export class CategoryGroupService { constructor (@InjectModel(Group.name) private categoryGroupModel: Model<IGroup>) {} ; async getAll(userId: UserIdSession): Promise<Group[]> { const userAllCategories = this.categoryGroupModel.find({ user: userId }, PROJECTIONS); return await userAllCategories.lean(); } ; async add (@Body() createCategoryGroupDto: CreateCategoryGroupDto, userId: UserIdSession) { const dataForAdd = { user: userId, name: createCategoryGroupDto.name }; const newCategoryGroup = await new this.categoryGroupModel(dataForAdd); const newCategoryGroupDocument = await newCategoryGroup.save(); return newCategoryGroupDocument; } ; async delete(id: string, userId: UserIdSession): Promise<unknown> { return this.categoryGroupModel.deleteOne({ _id: id, user: userId }); } ; async edit(id: string, body: UpdateCategoryGroupDto, userId: UserIdSession) { const updatedCategoryGroup = await this.categoryGroupModel.findByIdAndUpdate({ _id: id, user: userId }, body, { new: true }); if (!updatedCategoryGroup) { return null; } const updatedCategoryGroupDocument = await updatedCategoryGroup.save(); return updatedCategoryGroupDocument; } }

```

src/category-group/dto/create-category-group.dto.ts

```

import { ApiProperty } from '@nestjs/swagger'; import { IsNotEmpty, IsString } from 'class-validator'; export class CreateCategoryGroupDto { @ApiProperty() @IsNotEmpty() @IsString(); name: string; }

```

src/category-group/dto/update-category-group.dto.ts

```

import { PartialType } from '@nestjs/mapped-types'; import { CreateCategoryGroupDto } from './create-category-group.dto'; export class UpdateCategoryGroupDto extends PartialType(CreateCategoryGroupDto) {}

```

src/color/color.config.ts

```

import { ColorsNames } from './color.interface'; import { Color } from './color.schema'; export const DEFAULT_COLORS: Color[] = [ { name: ColorsNames.RED, colorHEX: '#ef4444', order: 1 }, { name: ColorsNames.FUCHSIA, colorHEX: '#d946ef', order: 2 }, { name: ColorsNames.BLUE, colorHEX: '#3b82f6', order: 3 }, { name: ColorsNames.CYAN, colorHEX: '#06b6d4', order: 4 }, { name: ColorsNames.EMERALD, colorHEX: '#10b981', order: 5 }, { name: ColorsNames.LIME, colorHEX: '#84cc16', order: 6 }, { name: ColorsNames.GRAY, colorHEX: '#6b7280', order: 7 }, { name: ColorsNames.AMBER, colorHEX: '#f59e0b', order: 8 } ]

```

src\color\color.controller.ts

```
import { Controller, Get, Headers, HttpStatus, Post, Res } from '@nestjs/common';import { ColorService } from '../color.service';import { Color } from '../color.schema';import { Response } from 'express';import { ConfigService } from '@nestjs/config'@Controller('color');export class ColorController { constructor(private colorService: ColorService, private configService: ConfigService) { } @Get('all'); async getColors(): Promise<Color[]> { return this.colorService.getAll() } @Post('initialize'); async initializeDefaultColors(@Headers('authorization') authorization: undefined | string, @Res() res: Response) { console.log(authorization); if (!authorization) { res.status(HttpStatus.UNAUTHORIZED).json({ message: 'UNAUTHORIZED' }); return }; const INITIALIZE_LOGIN = this.configService.get<string>('INITIALIZE_LOGIN'); const INITIALIZE_PASSWORD = this.configService.get<string>('INITIALIZE_PASSWORD'); console.log(`${INITIALIZE_LOGIN}:${INITIALIZE_PASSWORD}`); if (authorization !== `${INITIALIZE_LOGIN}:${INITIALIZE_PASSWORD}`) { res.status(HttpStatus.UNAUTHORIZED).json({ message: 'The login or password is incorrect' }); return }; return res.status(HttpStatus.OK).json(await this.colorService.initializeDefaultColors()) }
```

src\color\color.interface.ts

```
import { HydratedDocument } from 'mongoose';import { Color } from '../color.schema';export const enum ColorsNames { GRAY = 'gray', RED = 'red', AMBER = 'amber', LIME = 'lime', EMERALD = 'emerald', CYAN = 'cyan', BLUE = 'blue', FUCHSIA = 'fuchsia' };export type IColor = HydratedDocument<Color>
```

src\color\color.module.ts

```
import { Module } from '@nestjs/common';import { ColorController } from '../color.controller';import { MongooseModule } from '@nestjs/mongoose';import { Color, ColorSchema } from '../color.schema';import { ColorService } from '../color.service';import { ConfigModule } from '@nestjs/config'@Module({ imports: [ConfigModule, MongooseModule.forFeature([ { name: Color.name, schema: ColorSchema } ])], controllers: [ColorController], providers: [ColorService], exports: [ColorService] });export class ColorModule { }
```

src\color\color.schema.ts

```
import { Prop, Schema, SchemaFactory } from '@nestjs/mongoose';import { ColorsNames } from '../color.interface'@Schema();export class Color { @Prop({ required: true }); name: ColorsNames @Prop({ required: true }); colorHEX: string @Prop({ required: true }); order: number };export const ColorSchema = SchemaFactory.createForClass(Color)
```

src\color\color.service.ts

```
import { Body, Injectable } from '@nestjs/common';import { InjectModel } from '@nestjs/mongoose';import { Model, ObjectId } from 'mongoose';import { DEFAULT_COLORS } from './
```

```

color.config';import{ ColorsNames,IColor }from'./color.interface';import{ Color }from'./c
olor.schema';import{ CreateColorDto }from'./dto/create-
color.dto';import{ UpdateColorDto }from'./dto/update-color.dto'@Injectable();export
class ColorService{ config={ projection:`-__v-createdAt-
updatedAt` };constructor( @InjectModel(Color.name)private
colorModel:Model<IColor>){ };async
create( @Body()createColorDto:CreateColorDto):Promise<IColor>{ const
newColor=await new this.colorModel(createColorDto);return newColor.save();}async
update(colorId:string,updateColorDto:UpdateColorDto):Promise<IColor>{ const
existingColor=await
this.colorModel.findByIdAndUpdate(colorId,updateColorDto,{ new:true});if(!existingC
olor){ throw new Error(`Color#${colorId}not found`)};return existingColor};async
find(data:Partial<Color>):Promise<IColor|null>{ return await
this.colorModel.findById(data,this.config.projection).lean();}async
getAll():Promise<IColor[]>{ return await
this.colorModel.find({ },this.config.projection).lean();}async
delete(colorId:string){ const deletedColor=await
this.colorModel.findByIdAndDelete(colorId);if(!deletedColor){ throw new
Error(`Student#${colorId}not found`)};return deletedColor};async
initializeDefaultColors(){ const allColors=await
this.getAll();this.createOrUpdateColors(allColors,DEFAULT_COLORS);this.removeU
nUsedColors(allColors,DEFAULT_COLORS);this.removeDuplicates(allColors);const
colorItems=await this.getAll();const
resultObject:any={ };colorItems.forEach(item=>{ if(item._id){ resultObject[item.name]=
item._id.toString() });return resultObject as{ [key in ColorsNames]:string };private
async
createOrUpdateColors(all:IColor[],defaults:Color[]){ defaults.forEach(defaultColor=>{ c
onst
findedColor=all.find(color=>{ return(defaultColor.name===color.name||defaultColor.co
lorHEX===color.colorHEX||defaultColor.order===color.order)});if(findedColor===un
defined){ this.create(defaultColor);return };if(findedColor.name!==defaultColor.name||fi
ndedColor.colorHEX!==defaultColor.colorHEX||findedColor.order!==defaultColor.ord
er){ this.update(findedColor._id.toString(),defaultColor);return } } });private async
removeUnUsedColors(all:IColor[],defaults:Color[]){ all.forEach(color=>{ const
findedDefaultColor=defaults.find(defaultColor=>{ return(defaultColor.name===color.n
ame||defaultColor.colorHEX===color.colorHEX)});if(findedDefaultColor===undefine
d){ this.delete(color._id.toString());return } } });private async
removeDuplicates(all:IColor[]){ const cache:Record<string,string>={ };const
forDelete:IColor[]=[];all.forEach(item=>{ if(item.name in
cache){ forDelete.push(item);return };cache[item.name]=item.colorHEX });forDelete.for
Each(color=>{ this.delete(color._id.toString()) } ) }

```

src\color\dto\create-color.dto.ts

```
import { ApiProperty } from '@nestjs/swagger';import { IsNotEmpty,IsNumber,IsString } from 'class-validator';import { ColorsNames } from '../color.interface';export class CreateColorDto { @ ApiProperty()@ IsString()@ IsNotEmpty();readonly name:ColorsNames@ ApiProperty()@ IsString()@ IsNotEmpty();readonly colorHEX:string@ ApiProperty()@ IsNumber()@ IsNotEmpty();readonly order:number }
```

```
src\color\dto\update-color.dto.ts
```

```
import { PartialType } from '@nestjs/mapped-types';import { CreateColorDto } from './create-color.dto';export class UpdateColorDto extends PartialType(CreateColorDto){ }
```

```
src\config\constants.config.ts
```

```
export const COOKIE_MAX_AGE=2*360*24*60*60*1000
```

```
src\config\environment.config.ts
```

```
const
```

```
environmentConfig=()=>>({isGlobal:true,NODE_ENV:process.env.NODE_ENV,port:parseInt(process.env.PORT,10)||3000,database:{uri:process.env.MONGODB_URI},MONGODB_URI:process.env.MONGODB_URI});export {environmentConfig}
```

```
src\helpers\getHttpsOptions.helper.ts
```

```
import fs from'node:fs';import path from'node:path';const
```

```
getHttpsOptions=()=>>{return {key:fs.readFileSync(path.resolve(__dirname,'../certs/selfsigned.key')),cert:fs.readFileSync(path.resolve(__dirname,'../certs/selfsigned.crt'))}}};export {getHttpsOptions}
```

```
src\initialize\initialize.controller.ts
```

```
import { Controller,Get,HttpStatus,Res,Session } from '@nestjs/common';import { Response } from'express';import { SessionData } from'express-session';import { InitializeService } from'./initialize.service'@ Controller('initialize');export class InitializeController{ constructor(private initializeService:InitializeService){ } @ Get();async initialize( @ Session()session:SessionData, @ Res()res:Response){res.status(HttpStatus.OK).json(await this.initializeService.initialize(session))} }
```

```
src\initialize\initialize.module.ts
```

```
import { Module } from '@nestjs/common';import { UserModule } from'src/user/user.module';import { InitializeController } from'./initialize.controller';import { InitializeService } from'./initialize.service'@ Module( { imports:[UserModule],controllers:[InitializeController],providers:[InitializeService]});export class InitializeModule{ }
```

```
src\initialize\initialize.service.ts
```

```
import { Injectable } from '@nestjs/common';import { SessionData } from'express-session';import { UserService } from'../user/user.service'@ Injectable();export class InitializeService{ constructor(private userService:UserService){ };async
```

```
initialize(session:SessionData):Promise<unknown>{ if(!session.auth?.authorized||!session.auth?.userId){ return{ authorized:false } };const userData=await this.userService.find({ _id:session.auth.userId});return{ authorized:true,...userData } } }
```

```
src\statistic\statistic.config.ts
export const PROJECTIONS=`-__v-createdAt-updatedAt`
```

```
src\statistic\statistic.controller.ts
import{ Body,Controller,Delete,Get,HttpStatus,Param,Post,Put,Res,Session }from'@nestjs/common';import{ Response }from'express';import{ SessionData }from'express-session';import{ CreateStatisticDto }from'./dto/create-statistic.dto';import{ UpdateStatisticDto }from'./dto/update-statistic.dto';import{ StatisticService }from'./statistic.service';@Controller('statistic');export class StatisticController{ constructor(private statisticService:StatisticService){ } @Get('all');async getAll(@Session()session:SessionData,@Res()res:Response){ if(session.auth===undefined||!session.auth.authorized){ res.status(HttpStatus.UNAUTHORIZED).json({ message:'Unauthorized' });return };res.status(HttpStatus.OK).json(await this.statisticService.getAll(session.auth.userId)) } @Get('/:id');async getById(@Param('id')id:string,@Session()session:SessionData,@Res()res:Response){ if(session.auth===undefined||!session.auth.authorized){ res.status(HttpStatus.UNAUTHORIZED).json({ message:'Unauthorized' });return };res.status(HttpStatus.OK).json(await this.statisticService.getById(id,session.auth.userId)) } @Post('add');async add(@Body()body:CreateStatisticDto,@Session()session:SessionData,@Res()res:Response){ if(session.auth===undefined||!session.auth.authorized){ res.status(HttpStatus.UNAUTHORIZED).json({ message:'Unauthorized' });return };res.status(HttpStatus.OK).json(await this.statisticService.add(body,session.auth.userId)) } @Delete('/:id');async delete(@Param('id')id:string,@Session()session:SessionData,@Res()res:Response){ if(session.auth===undefined||!session.auth.authorized){ res.status(HttpStatus.UNAUTHORIZED).json({ message:'Unauthorized' });return };res.status(HttpStatus.OK).json(await this.statisticService.delete(id,session.auth.userId)) } @Put('/:id');async put(@Param('id')id:string,@Body()body:UpdateStatisticDto,@Session()session:SessionData,@Res()res:Response){ if(session.auth===undefined||!session.auth.authorized){ res.status(HttpStatus.UNAUTHORIZED).json({ message:'Unauthorized' });return };const result=await this.statisticService.edit(id,body,session.auth.userId);if(result===null){ return res.status(HttpStatus.BAD_REQUEST).json({ message:'Something gonna wrong' }) };res.status(HttpStatus.OK).json(result) } }
```

```
src\statistic\statistic.interface.ts
import{ HydratedDocument }from'mongoose';import{ Statistic }from'./statistic.schema';export type IStatistic=HydratedDocument<Statistic>
```

```
src\statistic\statistic.module.ts
```

```
import { Module } from '@nestjs/common'; import { StatisticController } from './statistic.controller'; import { StatisticService } from './statistic.service'; import { MongooseModule } from '@nestjs/mongoose'; import { Statistic, StatisticSchema } from './statistic.schema' @Module({ imports: [MongooseModule.forFeature([ { name: Statistic.name, schema: StatisticSchema } ])], controllers: [StatisticController], providers: [StatisticService] }); export class StatisticModule { }
```

src\statistic\statistic.schema.ts

```
import { Prop, Schema, SchemaFactory } from '@nestjs/mongoose'; import mongoose from 'mongoose'; import { Category } from 'src/category/category.schema'; import { User } from 'src/user/user.schema' @Schema(); export class Statistic { @Prop({ type: mongoose.Schema.Types.ObjectId, ref: 'User', required: true }); user: User @Prop({ required: true }); date: Date @Prop({ required: true }); count: number @Prop(); comment: string @Prop({ type: mongoose.Schema.Types.ObjectId, ref: 'Category', required: true }); category: Category @Prop({ required: true }); summ: number }; export const StatisticSchema = SchemaFactory.createForClass(Statistic)
```

src\statistic\statistic.service.ts

```
import { Injectable } from '@nestjs/common'; import { InjectModel } from '@nestjs/mongoose'; import { Model } from 'mongoose'; import { UserIdSession } from 'src/app.interfaces'; import { CreateStatisticDto } from './dto/create-statistic.dto'; import { UpdateStatisticDto } from './dto/update-statistic.dto'; import { PROJECTIONS } from './statistic.config'; import { IStatistic } from './statistic.interface'; import { Statistic } from './statistic.schema' @Injectable(); export class StatisticService { constructor(@InjectModel(Statistic.name) private statisticModel: Model<IStatistic> ) { }; async getAll(userId: UserIdSession) { const userAllStatistic = this.statisticModel.find({ user: userId }, PROJECTIONS).lean(); return await userAllStatistic.lean(); } async getById(id: string, userId: UserIdSession) { return await this.statisticModel.find({ _id: id, user: userId }, PROJECTIONS).lean(); } async add(body: CreateStatisticDto, userId: UserIdSession) { const dataForAdd = { user: userId, date: body.date, count: body.count, comment: body.comment, category: body.category, summ: body.summ }; const newStatistic = await this.statisticModel(dataForAdd); const newStatisticDocument = await newStatistic.save(); return newStatisticDocument; } async delete(id: string, userId: UserIdSession): Promise<unknown> { return await this.statisticModel.deleteOne({ _id: id, user: userId }); } async edit(id: string, body: UpdateStatisticDto, userId: UserIdSession) { const updatedStatistic = await this.statisticModel.findByIdAndUpdate({ _id: id, user: userId }, body, { new: true }); if (!updatedStatistic) { return null; } const updatedStatisticDocument = await updatedStatistic.save(); return updatedStatisticDocument; } }
```

src\statistic\dto\create-statistic.dto.ts


```
import { ApiProperty } from '@nestjs/swagger';import { IsNotEmpty, IsString } from 'class-validator';export class
CreateStatisticDto { @ApiProperty()@IsNotEmpty()@IsString();date:string@ApiProperty()@IsNotEmpty()@IsString();count:number@ApiProperty()@IsNotEmpty()@IsString();comment:string@ApiProperty()@IsNotEmpty()@IsString();category:string@ApiProperty()@IsNotEmpty()@IsString();summ:number}
```

```
src\statistic\dto\update-statistic.dto.ts
import { PartialType } from '@nestjs/mapped-types';import { CreateStatisticDto } from './create-statistic.dto';export class
UpdateStatisticDto extends PartialType(CreateStatisticDto){}
```

```
src\user\user.interface.ts
import { HydratedDocument } from 'mongoose';import { User } from './user.schema';export
type IUser=HydratedDocument<User>
```

```
src\user\user.module.ts
import { Module } from '@nestjs/common';import { UserService } from './user.service';import
{ MongooseModule } from '@nestjs/mongoose';import { User, UserSchema } from './user.schema';import { CategoryModule } from '../category/category.module'@Module({ imports:[CategoryModule,MongooseModule.forFeature([ { name:User.name,schema:UserSchema} ]),controllers:[],providers:[UserService],exports:[UserService]});export class
UserModule{}
```

```
src\user\user.schema.ts
import { Prop, Schema, SchemaFactory } from '@nestjs/mongoose'@Schema();export class
User { @Prop({ required:true });name:string@Prop({ required:true });picture:string@Prop({ required:true });email:string@Prop({ required:true });email_verified:boolean@Prop({ required:true });given_name:string@Prop({ required:true });family_name:string};export
const UserSchema=SchemaFactory.createForClass(User)
```

```
src\user\user.service.ts
import { Body, Injectable } from '@nestjs/common';import { InjectModel } from '@nestjs/mongoose';import
mongoose, { Model, ObjectId } from 'mongoose';import { IUser } from './user.interface';import
{ User } from './user.schema';import { CreateUserDto } from './dto/create-user.dto';import { CategoryService } from '../category/category.service'@Injectable();export
class
UserService { constructor( @InjectModel(User.name)private
userModel:Model<IUser>,private categoryService:CategoryService){};async
create( @Body()createUserDto:CreateUserDto,options: { initializeDefaultCategories:boolean }={ initializeDefaultCategories:false }):Promise<IUser>{ const newUser=await new
this.userModel(createUserDto);if(options.initializeDefaultCategories){ this.categoryService.initializeUserDefaultCategories(newUser._id)};return newUser.save()};async
```

```

find(data:Partial<User&{_id:string|ObjectId|mongoose.Types.ObjectId}>):Promise<IU
ser|null>{return await this.userModel.findOne(data).lean()};async
findOrCreate(data:User):Promise<IUser>{const
dataForFind={name:data.name,email:data.email,email_verified:data.email_verified,give
n_name:data.given_name,family_name:data.family_name};const findedUser=await
this.find(dataForFind);if(findedUser!==null){return findedUser};return await
this.create(data)}}

```

```

src\user\dto\create-user.dto.ts
import { ApiProperty } from '@nestjs/swagger';import { IsBoolean,IsEmail,IsNotEmpty,IsS
tring } from 'class-validator';export class
CreateUserDto { @ApiProperty()@IsString()@IsNotEmpty();name:string@ ApiProperty
()@IsString()@IsNotEmpty();picture:string@ ApiProperty()@IsEmail()@IsNotEmpty()
;email:string@ ApiProperty()@IsBoolean()@IsNotEmpty();email_verified:boolean@Ap
iProperty()@IsString()@IsNotEmpty();given_name:string@ ApiProperty()@IsString()
@IsNotEmpty();family_name:string}

```