МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет водного господарства та природокористування
Навчально-науковий інститут кібернетики, інформаційних технологій та інженерії
Кафедра комп'ютерних технологій та економічної кібернетики

Допущено до захисту:

Завідувач кафедри
комп'ютерних технологій та
економічної кібернетики
д. е. н., проф. П. М. Грицюк

_____

«____» _____ 2025 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня «магістр»
за освітньо-професійною програмою
«Інформаційні технології в бізнесі»
спеціальності 126 «Інформаційні системи та технології»
на тему: «Інформаційна система готельно-ресторанного бізнесу (на прикладі ГРК «Айвенго»)»

Виконала:
здобувач вищої освіти 2 курсу,
групи ІТБ-61м
Мельник Яна Василівна

Керівник:
к.т.н., доцент Василів В. Б.

Рецензент:
д.е.н., проф. Грицюк П. М.

Рівне – 2025

РЕФЕРАТ

Кваліфікаційна робота магістра: 70 с., 30 рис., 2 табл., 23 літературних джерел.

Актуальність теми магістерської роботи зумовлена необхідністю використання сучасних інформаційних технологій для автоматизації аналізу фінансових показників у готельно-ресторанному бізнесі. Застосування серверної платформи Node.js дозволяє створювати продуктивні та масштабовані інформаційні системи, що забезпечують оперативну обробку даних та формування аналітичних звітів у вебсередовищі.

Об’єктом дослідження є процес збору, зберігання та аналізу фінансово-економічних даних діяльності готельно-ресторанного комплексу.

Предметом дослідження є методи та технології розробки аналітичної веборієнтованої інформаційної системи з використанням середовища Node.js, системи керування базами даних MySQL та клієнтських технологій HTML, CSS і JavaScript.

Метою магістерської роботи є створення аналітичної інформаційної системи, що забезпечує введення фінансових показників, їх збереження у базі даних та формування наочних управлінських звітів у вигляді таблиць і графіків.

У магістерській роботі здійснено аналіз особливостей інформаційного забезпечення готельно-ресторанного бізнесу, обґрунтовано вибір технологічного стеку, спроєктовано структуру бази даних та реалізовано вебзастосунок для аналізу доходів, витрат і прибутку підприємства.

КЛЮЧОВІ СЛОВА: АНАЛІТИЧНА ІНФОРМАЦІЙНА СИСТЕМА, ГОТЕЛЬНО-РЕСТОРАННИЙ БІЗНЕС, ВЕБЗАСТОСУНОК, NODE.JS, EXPRESS, MYSQL, БАЗА ДАНИХ, JAVASCRIPT.

ЗМІСТ

ВСТУП

Сучасний готельно-ресторанний бізнес функціонує в умовах високої конкуренції, динамічних змін ринку та зростаючих вимог до ефективності управління. Одним із ключових чинників успішної діяльності підприємств даної сфери є наявність достовірної, структурованої та своєчасної аналітичної інформації щодо фінансово-економічних показників. Обсяги доходів, витрат та прибутку формуються щоденно, що ускладнює їх ручний облік та аналіз і підвищує ризик виникнення помилок при прийнятті управлінських рішень.

У зв’язку з цим актуальним є впровадження аналітичних інформаційних систем, здатних автоматизувати процеси збору, збереження та обробки фінансових даних, а також забезпечити наочне подання результатів у вигляді таблиць і графіків. Використання веборієнтованих технологій дозволяє створювати доступні та масштабовані рішення, які можуть бути використані управлінським персоналом незалежно від місця та часу доступу.

Особливу увагу в сучасних умовах привертає серверна платформа Node.js, яка забезпечує високу продуктивність, кросплатформеність та ефективну роботу з асинхронними операціями. Поєднання Node.js із системою керування базами даних MySQL та клієнтськими технологіями HTML, CSS і JavaScript створює передумови для розробки аналітичних вебзастосунків, орієнтованих на практичні потреби.

Метою магістерської роботи є проєктування та розробка аналітичної інформаційної системи для готельно-ресторанного комплексу «Айвенго», яка забезпечує введення щоденних фінансових показників, їх збереження у базі даних, автоматизоване агрегування та формування управлінських аналітичних звітів.

Об’єктом дослідження є процес інформаційно-аналітичного забезпечення управління фінансовою діяльністю готельно-ресторанного комплексу.

Предметом дослідження є методи та програмні засоби розробки веборієнтованої аналітичної інформаційної системи для обліку та аналізу доходів і витрат підприємства.

Для досягнення поставленої мети в магістерській роботі передбачено розв’язання таких завдань:

- проаналізувати особливості діяльності готельно-ресторанних комплексів та роль інформаційних систем у процесі управління;
- здійснити огляд сучасних інформаційних систем, що застосовуються у готельно-ресторанному бізнесі;
- обґрунтувати вибір технологічного стеку для розробки аналітичної інформаційної системи;
- спроєктувати структуру бази даних для зберігання фінансових показників;
- розробити серверну частину вебзастосунку для обробки та збереження даних;
- реалізувати клієнтську частину системи з можливістю візуалізації аналітичної інформації;
- провести тестування працездатності розробленої інформаційної системи.

Логіка дослідження зумовила структуру магістерської роботи, яка складається зі вступу, трьох розділів, висновків, списку використаних джерел та додатків. У першому розділі розглянуто теоретичні основи інформаційних систем готельно-ресторанного бізнесу та проаналізовано існуючі рішення. У другому розділі виконано проєктування аналітичної інформаційної системи, зокрема структури даних та архітектури вебзастосунку. Третій розділ присвячено програмній реалізації системи, опису її функціональних можливостей та аналізу отриманих результатів.

Під час виконання магістерської роботи використано такі програмні засоби та технології: Visual Studio Code, Node.js, Express, MySQL, MySQL Workbench, HTML, CSS, JavaScript, Chart.js та Microsoft Word.

РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ ІНФОРМАЦІЙНИХ СИСТЕМ ГОТЕЛЬНО-РЕСТОРАННОГО БІЗНЕСУ

1.1 Особливості діяльності готельно-ресторанних комплексів та роль інформаційних систем

Готельно-ресторанний бізнес посідає важливе місце у сфері послуг, оскільки поєднує надання житлових і харчових послуг, організацію дозвілля та створення комфортних умов для гостей. Діяльність готельно-ресторанних комплексів характеризується високою динамічністю, сезонністю попиту, значною кількістю щоденних операцій та необхідністю швидкого реагування на зміни потреб клієнтів. У таких умовах ефективне управління можливе лише за умови використання сучасних інформаційних систем.

Сучасний розвиток галузі HoReCa безпосередньо пов’язаний із впровадженням інформаційних технологій, які забезпечують автоматизацію ключових бізнес-процесів. До таких процесів належать бронювання номерів, облік замовлень у ресторані, управління складськими запасами, ведення бухгалтерського обліку, формування фінансової звітності та підтримка комунікації з клієнтами. Використання інформаційних систем дозволяє зменшити вплив людського фактору, скоротити час обробки інформації та підвищити точність даних.

Інформаційні системи у готельно-ресторанних комплексах виконують роль інтеграційної платформи, яка об’єднує різні підрозділи підприємства в єдиний інформаційний простір. Завдяки цьому керівництво отримує можливість оперативно контролювати фінансові показники, аналізувати доходи та витрати, оцінювати ефективність роботи окремих підрозділів і приймати обґрунтовані управлінські рішення.

Застосування інформаційних систем надає готелям та ресторанам можливість:

- централізовано управляти процесами бронювання та обслуговування клієнтів;
- здійснювати бухгалтерський, фінансовий та складський облік у режимі реального часу;
- формувати аналітичні та управлінські звіти для оцінки результатів діяльності;
- забезпечувати інтеграцію з зовнішніми інформаційними сервісами, зокрема онлайн-платформами бронювання (Booking.com, Expedia тощо).

Особливе значення для управління готельно-ресторанним комплексом має аналітична складова інформаційних систем. Саме аналітичні інструменти дозволяють узагальнювати великі обсяги даних, виявляти тенденції, прогнозувати фінансові результати та оцінювати рентабельність діяльності підприємства. Відсутність або недостатній рівень автоматизації аналітичних процесів ускладнює прийняття стратегічних рішень та може призводити до зниження конкурентоспроможності.

Актуальність використання інформаційних систем у сфері HoReCa підтверджується результатами численних наукових досліджень, у яких підкреслюється, що ефективність діяльності підприємств залежить не лише від якості сервісу, але й від рівня розвитку їх інформаційно-технологічної інфраструктури. Таким чином, впровадження аналітичних інформаційних систем є необхідною умовою сталого розвитку готельно-ресторанних комплексів у сучасних економічних умовах.

1.2. Огляд і класифікація сучасних інформаційних систем у готельному та ресторанному бізнесі

Інформаційні системи у готельно-ресторанному бізнесі включають програми для бронювання номерів, управління ресторанними замовленнями,

контролю персоналу та управління фінансами. Вони можуть бути локальними або хмарними (доступними через інтернет-браузер).

У науковій літературі зазначається, що основними критеріями ефективності інформаційних систем є:

- узгодженість даних між різними модулями;
- зручність інтерфейсу для персоналу;
- інтеграція з міжнародними каналами бронювання;
- можливість швидкого формування звітності.

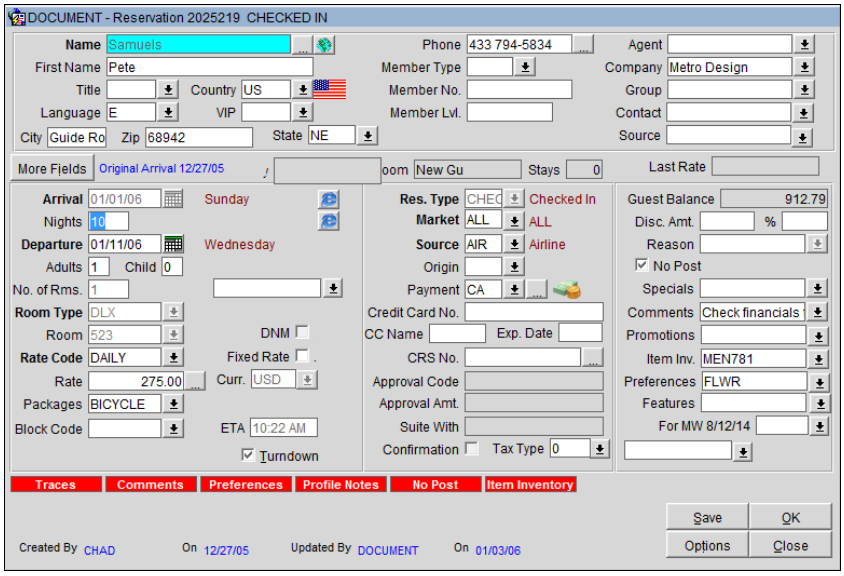


Рис. 1.1 Інтерфейс корпоративної системи управління (MS Opera)

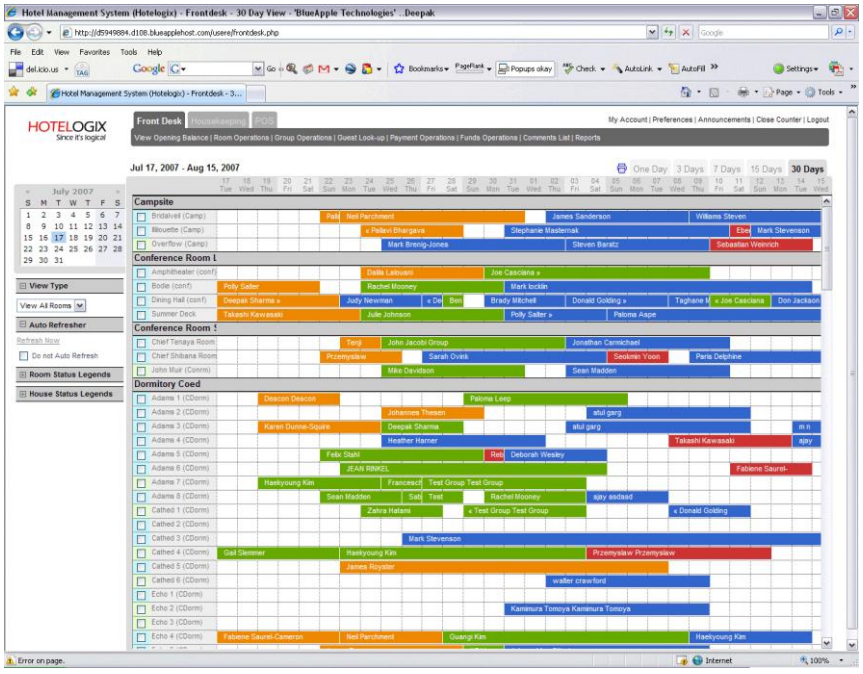


Рис. 1.2 Інтерфейс хмарної PMS Hotelogix

Серед найпоширеніших PMS-систем у світі можна назвати **MS Opera, Hotelogix** та локальні рішення на кшталт **B52**.

- Hotelogix орієнтований на невеликі й середні готелі, має хмарну архітектуру та простий у використанні інтерфейс.
- MS Opera – корпоративний стандарт, який використовується великими готельними мережами. Він має широкий спектр функцій, але вимагає значних витрат на впровадження та навчання персоналу.
- B52 – українська PMS, що поєднує в собі модулі готелю та ресторану і адаптована до вимог місцевих закладів.

Таблиця 1.1

Порівняння PMS-систем

Критерій	Hotelogix	OPERA	B52
Тип розгортання	Хмарна PMS (SaaS), веб-доступ	класично – локальне/гібридне	Зазвичай локальне/гібридне
Цільовий сегмент	Малі та середні	Середні та великі мережі	Малі та середні заклади
Інтеграції	OTA/метапошук, платежі, базові API	POS, CRM, RMS, OTA, платежі, безпека	POS/бухгалтерія; кастомні обміни (CSV/JSON)
Аналітика/звітність	Стандартні дашборди	Розширена аналітика, кастомні звіти	Типові звіти
Мультимовність/ валюта	Підтримується	Підтримується	Підтримується
Простота впровадження	Швидкий старт	Потребує навчання	Швидко в межах підприємства
Вартість	Передплата	Вища	Залежить від конфігурації
Сильні сторони	Хмара, простота, OTA-інтеграції	Повний функціонал, масштабованість	Локальна адаптація (готель+ресторан +бухгалтерія)
Обмеження	Менше глибоких налаштувань	Складність і вартість	Менше готових інтеграцій з глобальними сервісами

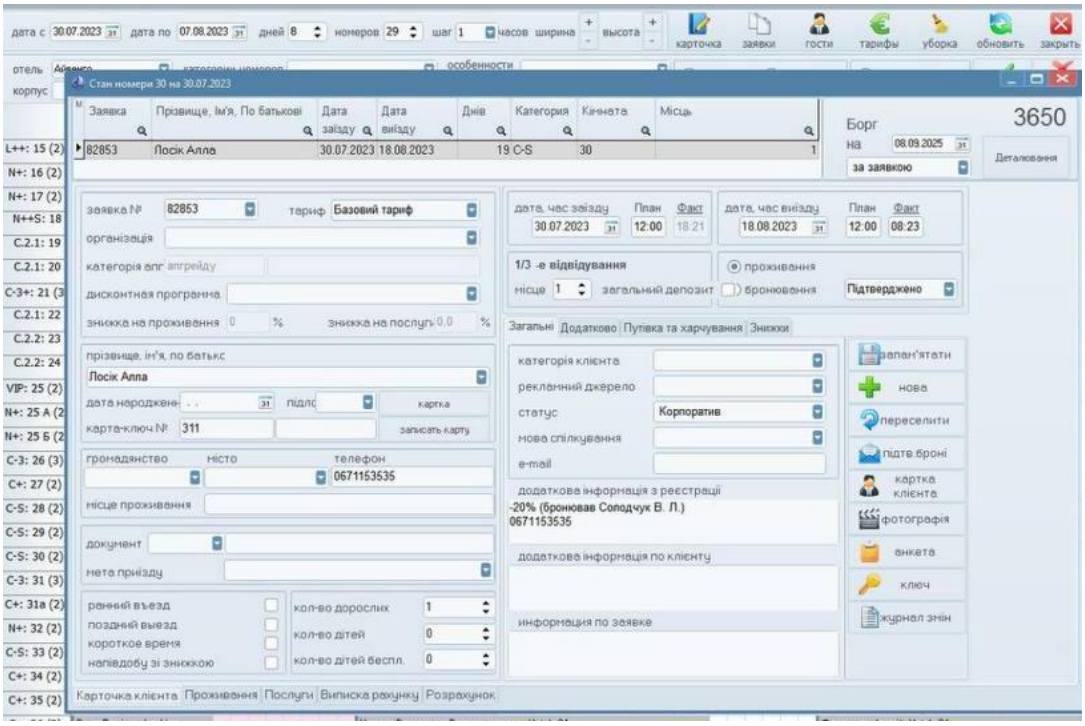


Рис. 1.3 Інтерфейс B52

Порівняння PMS-систем (табл. 1.1) та проілюстровані інтерфейси (рис. 1.1–1.3) показують, що хмарні рішення на кшталт *Hotelogix* забезпечують швидкий старт і інтеграції з каналами продажів, корпоративна *OPERA* надає найповніший функціонал за вищої складності впровадження, а локальні системи типу *B52* краще узгоджуються з процесами вітчизняних закладів, хоча потребують налаштування інтеграцій із зовнішніми сервісами.

Ключовим висновком є необхідність вибору такої ІС, яка не лише відповідає масштабу підприємства, а й забезпечує узгодженість даних між веб-сайтом, модулем бронювання та обліковими підсистемами.

1.3. Аналіз системи B52, що використовується в ГРК «Айвенго», та її функціональні можливості

Одним із ключових інструментів автоматизації управління діяльністю готельно-ресторанних закладів є програмний комплекс B52. Це вітчизняна

PMS-система (Property Management System), яка інтегрує основні бізнес-процеси: бронювання номерів, облік коштів, управління персоналом та планування додаткових послуг. Її використання забезпечує ефективну організацію роботи підприємства, скорочує час на обробку даних та зменшує кількість помилок, пов’язаних із ручним введенням інформації.

У ході практики було вивчено роботу окремих модулів B52, які активно застосовуються в управлінні:

1. Система автоматично фіксує час приходу та завершення зміни працівників, що дає змогу вести точний облік робочого часу і формувати звіти для бухгалтерії (рис. 1.4).

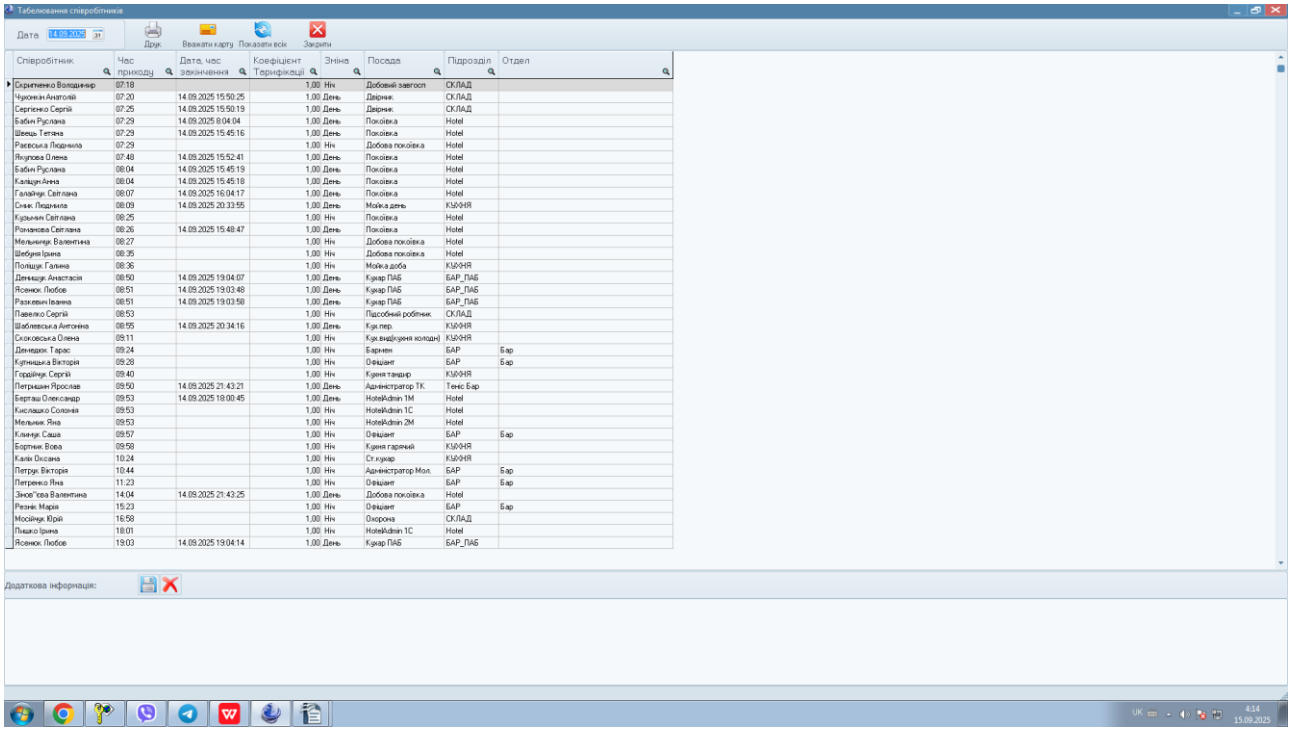


Рис. 1.4 Табелювання персоналу

2. До складу системи входять різні підсистеми: «Карта зайнятості», «Реєстр реєстрацій», «Планування поселення», «Каса», «Журнал повідомлень» тощо. Така структура забезпечує зручність переходу між різними операціями (рис. 1.5).

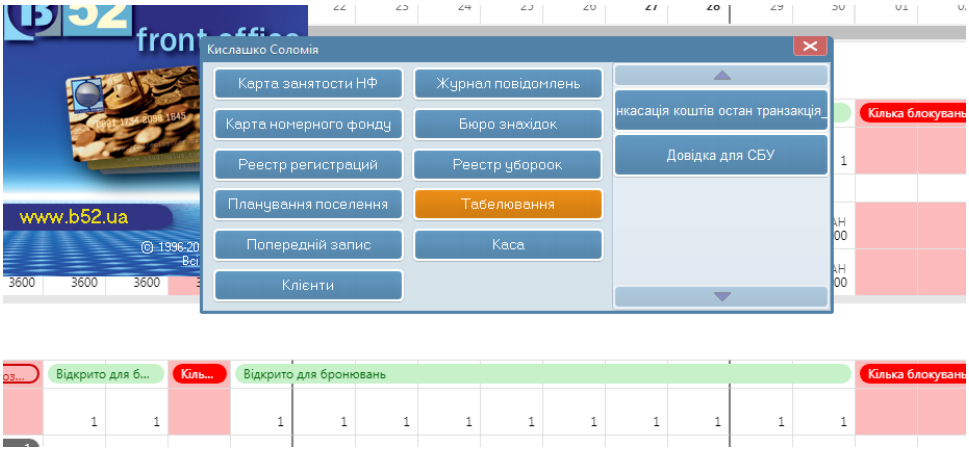


Рис. 1.5 Меню модулів B52

3. Наступний модуль дозволяє в реальному часі переглядати завантаженість номерів, що особливо важливо під час пікових періодів бронювання. Інформація відображається у вигляді календаря з вказаними боргами клієнтів і термінами перебування (рис. 1.6).

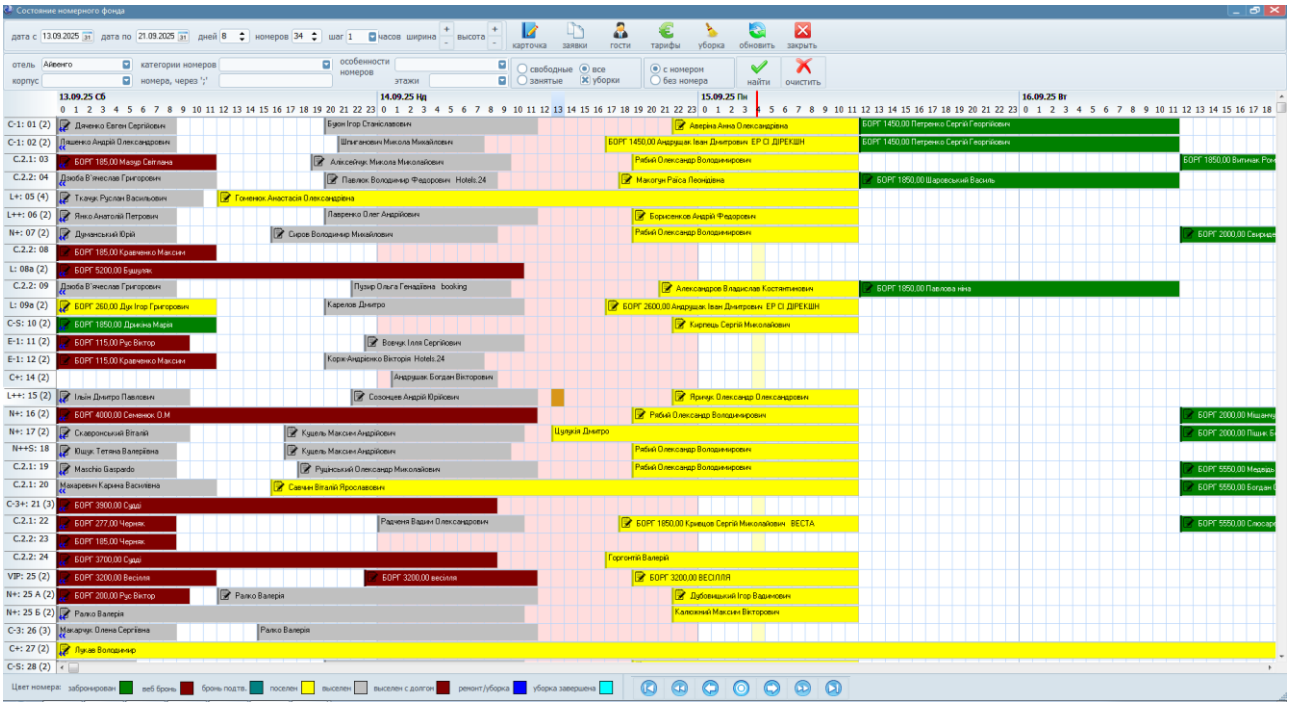


Рис. 1.6 Карта зайнятості номерного фонду

4. B52 забезпечує контроль касових операцій, ведення X- та Z-звітів, реєстрацію оплат готівкою та банківськими картками. Це дозволяє своєчасно формувати фінансову звітність і контролювати стан розрахунків із клієнтами (рис.1.7).

Рис. 1.7 Облік грошових коштів

5. Окремий модуль системи дає можливість створювати бронювання для таких сервісів, як більярд, сауна, конференц-зали чи ресторанні замовлення. Це дозволяє централізовано управляти не лише проживанням гостей, а й усіма видами додаткових послуг (рис. 1.8).

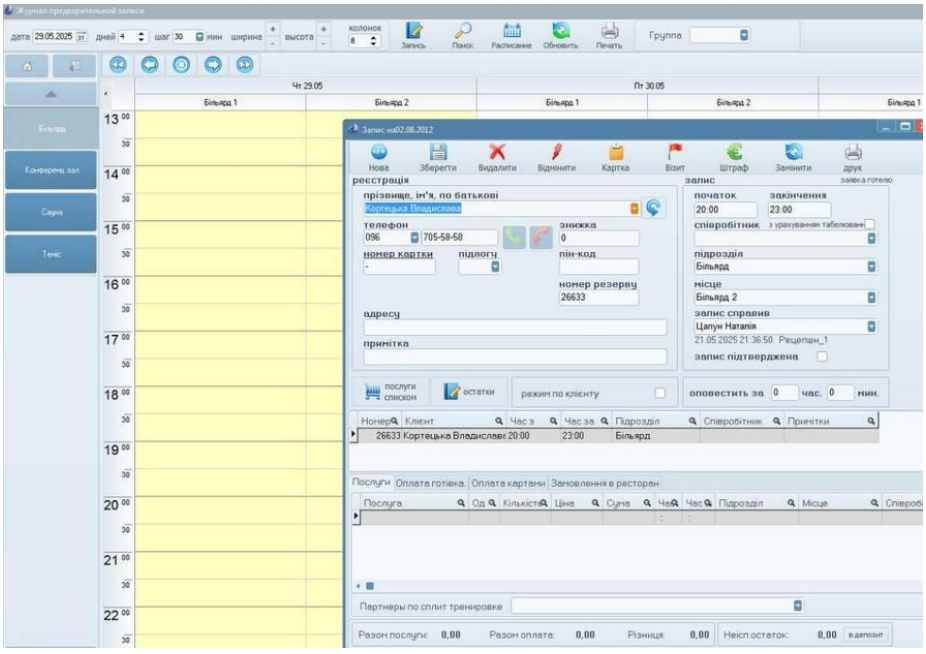


Рис. 1.8 Попередні записи та додаткові послуги

1.4. Проблеми та обмеження існуючих рішень і обґрунтування необхідності додаткової аналітичної підсистеми

Незважаючи на активне впровадження інформаційних систем у сфері готельно-ресторанного бізнесу, більшість сучасних програмних рішень зосереджені переважно на автоматизації операційної діяльності. Такі системи забезпечують облік бронювань, касових операцій, замовлень у ресторані, складських запасів і формування стандартної фінансової звітності. Водночас управлінська діяльність вимагає не лише фіксації подій, але й глибокого аналізу фінансових показників, виявлення тенденцій та підтримки процесу прийняття рішень.

Однією з ключових проблем існуючих інформаційних систем є роз’єднаність даних між різними підсистемами. Інформація про доходи, витрати та результати діяльності часто зберігається в окремих модулях або навіть у різних програмних продуктах, що ускладнює її узагальнення та аналіз. У результаті керівництво підприємства змушене виконувати додаткові ручні операції з об’єднання даних, що призводить до втрати часу та підвищує ймовірність помилок.

Ще одним суттєвим обмеженням є недостатній рівень аналітичних можливостей більшості існуючих рішень. Аналітика часто обмежується формуванням статичних звітів за фіксованими шаблонами, які не дозволяють гнучко змінювати період аналізу, порівнювати показники за різні часові інтервали або детально досліджувати структуру витрат. Такий підхід не відповідає потребам сучасного управління, де важливо оперативно отримувати відповіді на запити щодо причин змін фінансових результатів.

Важливою проблемою також є складність використання аналітичних модулів у комплексних інформаційних системах. Значна кількість функцій, налаштувань і технічних параметрів ускладнює роботу з аналітичними інструментами для управлінського персоналу, який не завжди має достатній

рівень ІТ-підготовки. Як наслідок, потенціал аналітичних можливостей таких систем використовується лише частково або залишається нереалізованим.

Крім того, багато сучасних інформаційних систем для готельно-ресторанного бізнесу є комерційними продуктами з високою вартістю впровадження та супроводу. Для малих і середніх підприємств це часто стає серйозним бар’єром, що обмежує доступ до сучасних аналітичних інструментів. У таких умовах аналіз фінансових показників здійснюється за допомогою спрощених засобів, зокрема електронних таблиць, які не забезпечують належного рівня автоматизації, безпеки та цілісності даних.

Окремої уваги заслуговує проблема своєчасності отримання аналітичної інформації. В умовах динамічного ринку затримка в отриманні даних про фінансові результати може призводити до прийняття неефективних управлінських рішень. Відсутність інструментів для аналізу даних у режимі наближеному до реального часу знижує здатність підприємства оперативно реагувати на зміни попиту, сезонні коливання та внутрішні фінансові ризики.

Зазначені проблеми свідчать про доцільність і необхідність розробки додаткової аналітичної підсистеми, орієнтованої на потреби конкретного готельно-ресторанного комплексу. Така підсистема повинна забезпечувати централізоване зберігання фінансових даних, автоматизоване агрегування показників за днями, місяцями та роками, а також підтримку порівняльного аналізу різних періодів діяльності.

Особливе значення при проєктуванні аналітичної підсистеми має наочність представлення результатів аналізу. Використання графіків, діаграм і зведених таблиць дозволяє швидко оцінити фінансовий стан підприємства, виявити ключові статті витрат і визначити фактори, що впливають на формування прибутку. Такий підхід підвищує ефективність управлінських рішень і сприяє стратегічному розвитку підприємства.

Для систематизації виявлених проблем і обмежень існуючих інформаційних систем у готельно-ресторанному бізнесі доцільно узагальнити результати аналізу у вигляді порівняльної таблиці. Такий підхід дозволяє

наочно відобразити взаємозв'язок між характерними недоліками сучасних рішень, їх впливом на управлінську діяльність підприємства та вимогами до додаткової аналітичної підсистеми. Узагальнення у табличній формі сприяє більш чіткому обґрунтуванню необхідності розробки аналітичної інформаційної системи, орієнтованої на потреби готельно-ресторанного комплексу.

Таблиця 1.2

Проблеми існуючих рішень

Проблема/обмеження	**Як проявляється в HoReCa**	**Управлінські наслідки**	**Що має дати аналітична підсистема (вимоги)**
Роз'єднані системи (PMS/POS/CRM/бронювання)	Показники зберігаються в різних місцях; виникають різні «версії правди»	Неузгоджені цифри, ручні звірки, затримка звітів	Єдине сховище/логіка агрегації; узгоджені KPI;
Потреба в інтеграції та «real-time insights»	Дані є, але швидко отримати управлінську картину важко	Рішення приймаються із запізненням	Дашборд/звіти з оперативним оновленням; зручні фільтри періоду
Обмежена аналітика: «звітність» замість «аналітики»	Є підсумки, але немає пояснення причин/рекомендацій	Неможливо швидко знайти драйвери падіння/зростання	Порівняння періодів, тренди, аналіз відхилень; перехід від опису до пояснення
Бар'єри впровадження BI (інтеграція, точність, ресурси, компетенції)	Дані збираються вручну, якість «плаває», бракує навичок	Помилки в управлінських висновках; недовіра до цифр	Валідація даних; стандарти введення; простий UI; автоматизація зведень
Невдалі/дорогі BI-впровадження в індустрії (історично)	BI не використовується масово або використовується частково	Користувачі повертаються до Excel і «ручного контролю»	Підсистема має бути зрозумілою, доступною, з вимірним ефектом; мінімальний поріг входу
Операційний фокус без глибокої фінансової аналітики	Підрахунок доходів/витрат є, але деталізація/порівняння слабкі	Складно контролювати витрати й прибутковість	Агрегації по місяцях/роках, структура витрат, графіки, порівняння з попереднім періодом

Отже, розробка аналітичної інформаційної системи, яка доповнює існуючі операційні рішення готельно-ресторанного комплексу, є обґрунтованою з точки зору сучасних вимог до управління. Впровадження такої підсистеми дозволяє подолати наявні обмеження, підвищити прозорість фінансових процесів і забезпечити керівництво інструментами для прийняття ефективних управлінських рішень.

1.5. Обґрунтування вибору технологій для розробки інформаційної системи

Розробка аналітичної інформаційної системи для готельно-ресторанного комплексу передбачає реалізацію трьох взаємопов’язаних складових: (1) серверної частини для обробки запитів і бізнес-логіки, (2) надійного сховища даних для довготривалого зберігання та агрегації показників, (3) клієнтського інтерфейсу для введення даних і візуалізації аналітики. Вибір технологічного стеку має забезпечувати продуктивність, масштабованість, простоту супроводу, можливість швидкого внесення змін у функціонал, а також доступність інструментів для розробника і кінцевого користувача. З урахуванням зазначених вимог у роботі обрано стек Node.js + Express + MySQL для серверної частини та сховища даних, а також HTML/CSS/JavaScript + Chart.js для побудови інтерфейсу й аналітичних графіків.

Node.js є кросплатформеним середовищем виконання JavaScript, орієнтованим на подієву (event-driven) модель і неблокуюче введення-виведення (non-blocking I/O). Для вебзастосунків, що обробляють значну кількість коротких запитів (збереження даних, отримання агрегованих звітів, фільтрація за періодом), така архітектура є практично доцільною, оскільки дозволяє ефективно працювати з паралельними з’єднаннями без надмірних

витрат системних ресурсів. Це особливо важливо для інформаційних систем, які повинні забезпечувати швидку реакцію інтерфейсу на дії користувача.

Додатковою перевагою Node.js є використання **єдиної мови програмування** (JavaScript) як на клієнтській, так і на серверній стороні. Це спрощує розробку й тестування, зменшує кількість контекстних переключень між мовами, а також полегшує підтримку проєкту. Для академічного проєкту та практичного впровадження у невеликому/середньому підприємстві така уніфікація підвищує керованість коду та швидкість внесення змін.

З точки зору екосистеми, Node.js має розвинений менеджер пакетів npm, який надає доступ до перевірених бібліотек для роботи з вебзастосунками, криптографією, автентифікацією, доступом до баз даних і візуалізацією. Використання стандартних компонентів екосистеми дозволяє будувати рішення на основі широко застосовуваних практик, а отже – зменшувати ризики та підвищувати підтримуваність системи.

Для реалізації серверної частини в межах Node.js обрано фреймворк Express, який забезпечує мінімалістичну, але достатньо потужну модель маршрутизації (routing) та організації middleware. У контексті даної інформаційної системи Express дозволяє:

- реалізувати сторінки доступу (login, адміністративне меню, введення даних, звіти);
- створити REST-подібний API для обміну даними між інтерфейсом і сервером (запити на збереження та отримання аналітики);
- застосувати єдині механізми перевірки доступу до захищених сторінок;
- структуровано розділити логіку застосунку на маршрути, обробники й допоміжні модулі.

Важливим аргументом на користь Express є його гнучкість: він не нав’язує жорсткої архітектури, що дозволяє адаптувати структуру проєкту під конкретні потреби – від невеликого MVP до розширеної системи з додатковими модулями.

Для зберігання фінансових даних (дохід, витрати, прибуток) обрано MySQL, оскільки ці дані мають чітку табличну структуру, логічні зв’язки та потребують забезпечення цілісності. Реляційна модель є природною для задач обліку і звітності, оскільки:

- гарантує коректність даних за рахунок обмежень (первинні ключі, типи, NOT NULL, перевірені значення);
- дозволяє виконувати агрегування (SUM, GROUP BY) та фільтрацію за періодами без додаткових обчислень на клієнтській стороні;
- підтримує транзакції, що критично при збереженні набору щоденних записів – дані повинні або зберегтися повністю, або не зберегтися взагалі.

Застосування MySQL Workbench полегшує адміністрування, візуальний контроль структури таблиць, перегляд даних та виконання SQL-запитів для перевірки коректності роботи системи. Для доступу до MySQL у проєкті використано бібліотеку mysql2, яка забезпечує роботу через пул з’єднань (connection pool). Це дає змогу зменшити накладні витрати на створення з’єднань і підвищити продуктивність при багаторазових запитах.

Окремої уваги заслуговує використання SQL-представлень (VIEW) для формування агрегованих показників. У проєкті застосовано представлення для розрахунку сумарних доходів і витрат за місяцями та визначення прибутку. Перенесення частини аналітичних обчислень на рівень бази даних є обґрунтованим, оскільки:

- зменшує обсяг даних, що передаються між сервером і клієнтом;
- гарантує єдину логіку розрахунків (одна «версія правди» для всіх користувачів);
- спрощує серверну частину, оскільки API повертає вже готові показники.

Інтерфейс системи реалізовано з використанням HTML5, CSS3 та JavaScript, що забезпечує кросплатформеність і доступність у будь-якому сучасному браузері без встановлення додаткового програмного забезпечення. Такий підхід є доцільним для системи, що орієнтується на практичне

використання в умовах підприємства, де важлива простота розгортання та мінімальні вимоги до робочих місць.

JavaScript на клієнтській стороні використовується для:

- динамічного формування таблиці введення даних залежно від обраного місяця (врахування кількості днів у місяці);
- підготовки та відправлення даних на сервер через fetch-запити;
- відображення результатів у вигляді таблиць і графіків;
- реалізації фільтрації й порівняння періодів.

Використання окремого CSS-файлу зі спільною стилізацією підвищує цілісність дизайну та спрощує супровід: зміни в оформленні застосовуються одразу на всіх сторінках.

Для відображення результатів аналізу застосовано бібліотеку Chart.js, яка дозволяє будувати лінійні графіки та інші типи діаграм на основі даних, отриманих від сервера. Її використання є обґрунтованим з таких причин:

- інтегрується безпосередньо у клієнтську частину без складних налаштувань;
- забезпечує наочне представлення трендів доходів, витрат і прибутку;
- підтримує оновлення графіків при зміні фільтра періоду або режиму порівняння;
- дозволяє підвищити зрозумілість аналітичної інформації для користувача, що є критично важливим для управлінського рівня.

З точки зору практики управління, графічне подання даних зменшує час на інтерпретацію показників і дозволяє швидко виявляти відхилення, сезонні коливання та тенденції.

Для обмеження доступу до функцій введення та перегляду аналітики реалізовано механізм автентифікації адміністратора. Паролі зберігаються у базі даних у вигляді хешів із використанням bcrypt, що відповідає базовим принципам безпечного зберігання облікових даних. Контроль доступу здійснюється через перевірку підписаних cookie, що забезпечує простий і достатній для навчально-практичної системи механізм авторизації.

У якості інструментів розробки та тестування використано Visual Studio Code (розробка), MySQL Workbench (адміністрування БД та SQL-запити), а також локальне розгортання сервера Node.js, що дозволяє повноцінно перевіряти роботу системи в умовах, наближених до реального використання.

Отже, обраний технологічний стек (Node.js, Express, MySQL, HTML/CSS/JavaScript, Chart.js) є обґрунтованим для реалізації аналітичної інформаційної системи готельно-ресторанного комплексу. Він забезпечує продуктивність, зручність розробки, можливість масштабування та зрозуміле представлення аналітичної інформації, що відповідає вимогам підприємства до оперативного аналізу фінансових показників.

Висновки по розділу 1

У першому розділі магістерської роботи розглянуто теоретичні та прикладні аспекти використання інформаційних систем у готельно-ресторанному бізнесі. Проаналізовано особливості діяльності готельно-ресторанних комплексів, що характеризуються значною кількістю щоденних операцій, сезонністю попиту та високими вимогами до оперативності управлінських рішень. Встановлено, що ефективне функціонування підприємств сфери HoReCa без використання сучасних інформаційних систем є ускладненим і супроводжується підвищеним ризиком помилок в обліку та аналізі даних.

У результаті аналізу існуючих інформаційних систем виявлено, що більшість програмних рішень орієнтовані переважно на автоматизацію операційних процесів, тоді як аналітична складова часто має обмежений або допоміжний характер. Недостатня інтеграція даних, фрагментарність аналітичної інформації, складність використання аналітичних модулів і висока вартість впровадження комплексних систем знижують ефективність застосування таких рішень у практичній діяльності готельно-ресторанних комплексів.

Обґрунтовано доцільність розробки додаткової аналітичної підсистеми, яка забезпечує централізоване зберігання фінансових показників,

автоматизоване агрегування даних та формування наочних управлінських звітів. Показано, що використання аналітичних інструментів дозволяє підвищити прозорість фінансових процесів, своєчасно виявляти тенденції та підтримувати прийняття управлінських рішень на основі даних.

На підставі проведеного аналізу обґрунтовано вибір технологічного стеку для розробки інформаційної системи, який включає середовище виконання Node.js, фреймворк Express, систему керування базами даних MySQL та клієнтські вебтехнології HTML, CSS і JavaScript. Визначено, що поєднання зазначених технологій забезпечує продуктивність, масштабованість, зручність розробки та можливість реалізації ефективної аналітичної підсистеми з візуалізацією результатів.

Отримані результати теоретичного аналізу створюють підґрунтя для переходу до практичної частини магістерської роботи, присвяченої проєктуванню архітектури аналітичної інформаційної системи та її програмній реалізації.

РОЗДІЛ 2. ПРОЕКТУВАННЯ АНАЛІТИЧНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ ГРК «АЙВЕНГО»

2.1. Постановка задачі та вимоги до майбутньої системи

На основі результатів теоретичного аналізу, проведеного у першому розділі магістерської роботи, встановлено, що ефективне управління фінансово-господарською діяльністю готельно-ресторанного комплексу потребує використання спеціалізованої аналітичної інформаційної системи. Така система повинна забезпечувати не лише збереження первинних даних про доходи та витрати, але й можливість їх оперативного аналізу, узагальнення та наочного представлення для підтримки управлінських рішень.

Основною задачею даної магістерської роботи є проєктування та розробка аналітичної інформаційної системи, призначеної для збору, зберігання, обробки та аналізу фінансових показників готельно-ресторанного комплексу. Система повинна надавати можливість обліку щоденних доходів і витрат, формування скорочених і детальних звітів, а також візуалізації результатів діяльності за різними часовими періодами.

Для досягнення поставленої мети необхідно вирішити такі задачі:

- організувати введення щоденних фінансових даних із урахуванням різної кількості днів у місяцях;
- забезпечити підтримку двох режимів обліку: скороченого (лише доходи) та детального (доходи й витрати за статтями);
- реалізувати збереження даних у базі даних із забезпеченням їх цілісності;
- забезпечити формування аналітичних звітів за місяць, рік та тривалий період;
- реалізувати механізми порівняння фінансових показників за різні періоди;

- забезпечити наочну візуалізацію результатів аналізу у вигляді таблиць і графіків;
- обмежити доступ до системи шляхом автентифікації користувача.

Функціональні вимоги до системи

Майбутня інформаційна система повинна забезпечувати реалізацію таких основних функцій:

- автентифікацію адміністратора для доступу до функціоналу системи;
- введення та редагування щоденних даних про доходи готелю та ресторану;
- введення та редагування даних про основні статті витрат у детальному режимі;
- автоматичне формування таблиці введення даних відповідно до обраного місяця;
- збереження введених даних у базі даних;
- формування аналітичних звітів за місяцями та роками;
- обчислення сумарних доходів, витрат і прибутку;
- порівняння показників поточного періоду з попереднім;
- відображення результатів у табличній та графічній формах.

Окрім функціональних можливостей, система повинна відповідати таким нефункціональним вимогам:

- зручність використання – інтерфейс має бути інтуїтивно зрозумілим для користувачів без спеціальної технічної підготовки;
- продуктивність – система повинна забезпечувати швидку обробку запитів і побудову аналітичних звітів;
- надійність – збереження даних має відбуватися з урахуванням транзакційності та захисту від втрати інформації;
- безпека – доступ до системи повинен бути обмежений авторизованими користувачами, а облікові дані мають зберігатися у захищеному вигляді;
- масштабованість – структура системи повинна дозволяти розширення функціоналу та додавання нових аналітичних показників;

- кросплатформеність – система має бути доступною через веббраузер незалежно від операційної системи.

Обмеження та умови реалізації

Під час розробки системи враховуються такі обмеження:

- система орієнтована на використання в межах одного готельно-ресторанного комплексу;
- доступ до системи здійснюється лише для адміністратора;
- аналітика базується на даних, введених користувачем вручну;
- система розгортається у вебсередовищі з використанням відкритих і поширених технологій.

У результаті реалізації поставлених задач має бути створена інформаційна система, яка забезпечить централізований облік фінансових показників готельно-ресторанного комплексу, підвищить прозорість фінансової діяльності та надасть інструменти для оперативного аналізу й прийняття управлінських рішень.

2.2. Структура даних для аналітики

Ефективність аналітичної інформаційної системи значною мірою залежить від правильно спроєктованої структури даних, яка повинна забезпечувати коректне зберігання первинної інформації, можливість її агрегування та подальшого аналізу за різними часовими періодами. Для готельно-ресторанного комплексу особливо важливо забезпечити збереження детальних фінансових показників із можливістю формування як скорочених, так і розширених аналітичних звітів.

При проєктуванні структури бази даних у межах даної роботи було обрано реляційний підхід, що дозволяє забезпечити цілісність даних, логічні

зв’язки між сутностями та ефективне виконання аналітичних запитів. Структура даних розроблена з урахуванням таких вимог:

- збереження щоденних доходів і витрат без втрати деталізації;
- можливість агрегації показників за місяцями та роками;
- підтримка різних режимів обліку (скороченого та детального);
- забезпечення розширюваності структури у разі додавання нових статей витрат або аналітичних показників.

У межах інформаційної системи для аналітики готельно-ресторанного комплексу виділено такі основні сутності:

- користувачі системи (адміністратори);
- щоденні доходи;
- щоденні витрати;
- агреговані аналітичні показники.

Кожна із зазначених сутностей реалізована у вигляді окремої таблиці або логічного представлення бази даних, що забезпечує чіткий розподіл відповідальності між рівнями зберігання та обробки інформації.

Таблиця користувачів призначена для збереження облікових даних адміністратора, який має доступ до функцій введення та аналізу фінансової інформації. У таблиці зберігаються логін користувача та хеш пароля, що відповідає вимогам безпеки та запобігає збереженню конфіденційних даних у відкритому вигляді. Наявність окремої таблиці користувачів дозволяє у подальшому реалізувати багаторівневу систему доступу або розширити функціонал системи для декількох ролей.

Для обліку доходів готельно-ресторанного комплексу використовується таблиця щоденних доходів, у якій фіксуються фінансові надходження за кожен календарний день. Основним ідентифікатором запису є дата, що дозволяє уникнути дублювання даних і спрощує виконання аналітичних запитів.

Доходи зберігаються окремо за основними напрямами діяльності комплексу, зокрема готельним та ресторанним. Такий підхід дозволяє:

- аналізувати внесок кожного напряму у загальний дохід;

- порівнювати ефективність різних підрозділів;
- виконувати агрегування даних за будь-який часовий період.

Збереження доходів у розрізі днів забезпечує високу деталізацію даних і створює основу для подальшого аналізу сезонних та короткострокових коливань.

Для обліку витрат використовується окрема таблиця щоденних витрат, яка містить інформацію про основні статті витрат готельно-ресторанного комплексу. До них належать витрати на електроенергію, продукти, заробітну плату персоналу, обслуговування та інші операційні витрати.

Розділення доходів і витрат на різні таблиці є обґрунтованим з точки зору нормалізації даних та забезпечує гнучкість структури. Зокрема, це дозволяє використовувати скорочений режим обліку, коли витрати автоматично фіксуються як нульові значення, або детальний режим, у якому зберігається повна інформація за кожною статтею.

Використання дати як первинного ключа таблиці витрат забезпечує однозначний зв’язок із таблицею доходів та спрощує виконання об’єднаних аналітичних запитів.

Для формування аналітичних показників у системі використовуються SQL-представлення, які агрегують первинні дані з таблиць щоденних доходів і витрат. Представлення дозволяють обчислювати загальний дохід, сумарні витрати та прибуток за місяць або рік без необхідності виконання складних обчислень на рівні серверної чи клієнтської логіки.

Використання представлень має такі переваги:

- зменшує навантаження на серверну частину застосунку;
- забезпечує єдину логіку розрахунків для всіх аналітичних запитів;
- спрощує реалізацію API для отримання агрегованих даних;
- підвищує надійність і прозорість аналітичних результатів.

Зв’язки між таблицями бази даних побудовані на основі календарної дати, яка виступає ключовим атрибутом для об’єднання доходів і витрат. Такий підхід дозволяє легко виконувати агрегацію даних за вибраними

періодами, а також розширювати структуру без суттєвих змін у вже реалізованій логіці.

Таким чином, розроблена структура даних забезпечує надійну основу для реалізації аналітичної інформаційної системи готельно-ресторанного комплексу. Вона поєднує детальне збереження первинних фінансових показників із можливістю їх ефективного агрегування та аналізу, що дозволяє формувати інформативні управлінські звіти та підтримувати прийняття обґрунтованих управлінських рішень.

2.3. Модель даних: опис сутностей та їх взаємозв’язків

Проєктування моделі даних є ключовим етапом розробки інформаційної системи, оскільки саме модель даних визначає логічну структуру зберігання інформації, зв’язки між окремими сутностями та правила їх взаємодії. Для аналітичної інформаційної системи готельно-ресторанного комплексу модель даних повинна забезпечувати збереження фінансових показників із необхідним рівнем деталізації, а також можливість їх подальшого агрегування та аналізу за різними часовими періодами. Реляційна модель дозволяє забезпечити цілісність даних, уникнути дублювання інформації та ефективно виконувати аналітичні запити із використанням операцій агрегації.

У межах аналітичної інформаційної системи виділено такі основні сутності:

- Користувач (User)
- Щоденний дохід (DailyRevenue)
- Щоденні витрати (DailyExpenses)
- Аналітичні показники (MonthlyProfit)

Кожна з цих сутностей реалізована у вигляді окремої таблиці або логічного представлення бази даних.

Сутність «Користувач» призначена для збереження облікових даних адміністратора системи. Вона містить унікальний ідентифікатор користувача, логін та хеш пароля, а також службову інформацію про час створення облікового запису. Використання хешування паролів забезпечує базовий рівень захисту доступу до системи.

Наявність окремої сутності користувача дозволяє у подальшому масштабувати систему, зокрема додати кілька ролей доступу або вести аудит дій користувачів.

Сутність «Щоденний дохід» призначена для збереження інформації про фінансові надходження готельно-ресторанного комплексу за кожен календарний день. Основним атрибутом цієї сутності є дата, яка використовується як первинний ключ і забезпечує унікальність запису.

Доходи зберігаються окремо за основними напрямами діяльності – готельним і ресторанним. Такий підхід дозволяє:

- аналізувати структуру доходів;
- визначати внесок кожного підрозділу в загальний фінансовий результат;
- здійснювати порівняльний аналіз у динаміці.

Сутність «Щоденні витрати» містить інформацію про основні статті витрат готельно-ресторанного комплексу за кожен день. До них належать витрати на електроенергію, продукти, заробітну плату, обслуговування та інші операційні витрати.

Як і у випадку з доходами, дата використовується як первинний ключ, що забезпечує однозначний зв’язок між витратами та доходами за відповідний день. Така структура дозволяє зберігати витрати як у скороченому режимі, так і в детальному режимі з повною інформацією за кожною статтею.

Аналітична сутність реалізована у вигляді SQL-представлення, яке агрегує дані зі сутностей «Щоденний дохід» та «Щоденні витрати». Представлення формує показники сумарного доходу, загальних витрат і прибутку за місяць або рік.

Використання логічної аналітичної сутності дозволяє:

- уникнути дублювання агрегованих даних;
- забезпечити єдину методику розрахунків;
- зменшити навантаження на серверну частину застосунку.

Взаємозв’язки між основними сутностями побудовані на основі календарної дати, яка є спільним атрибутом для таблиць доходів і витрат. Такий підхід забезпечує логічну цілісність моделі даних і дозволяє легко виконувати об’єднання таблиць під час формування аналітичних запитів.

Сутність користувача логічно пов’язана із системою доступу, але не бере безпосередньої участі в аналітичних розрахунках. Це дозволяє розділити функції безпеки та обліку фінансових даних.

На основі описаних сутностей і зв’язків сформовано логічну модель даних, яка забезпечує:

- збереження первинних фінансових показників без втрати деталізації;
- можливість агрегування даних за різними періодами;
- розширюваність структури при додаванні нових статей витрат або напрямів доходів;
- ефективну підтримку аналітичних запитів.

Логічна схема моделі даних може бути представлена у вигляді ER-діаграми, що наочно відображає сутності, їх атрибути та взаємозв’язки.

Таким чином, розроблена модель даних забезпечує логічну та структуровану основу для функціонування аналітичної інформаційної системи готельно-ресторанного комплексу. Вона поєднує детальне збереження фінансових показників із можливістю їх подальшого аналізу, що створює передумови для ефективної реалізації аналітичних функцій системи.

2.4. ER-діаграма бази даних аналітичної системи

ER-діаграма бази даних є важливим інструментом візуалізації структури інформаційної системи, оскільки дозволяє наочно відобразити сутності, їх

атрибути та взаємозв'язки між ними. Для аналітичної інформаційної системи готельно-ресторанного комплексу «Айвенго» ER-діаграма була побудована на основі фактично реалізованої бази даних із використанням засобів MySQL Workbench у режимі зворотного проєктування.

Застосування режиму Reverse Engineer дозволяє отримати актуальну логічну модель бази даних безпосередньо з фізичної структури, що гарантує повну відповідність діаграми реальній реалізації системи та виключає логічні розбіжності між проєктною документацією і програмним кодом.

На першому етапі було виконано підключення до сервера керування базами даних MySQL із використанням локального підключення. У параметрах з'єднання було задано стандартний TCP/IP метод, адресу сервера, порт та облікові дані користувача бази даних.

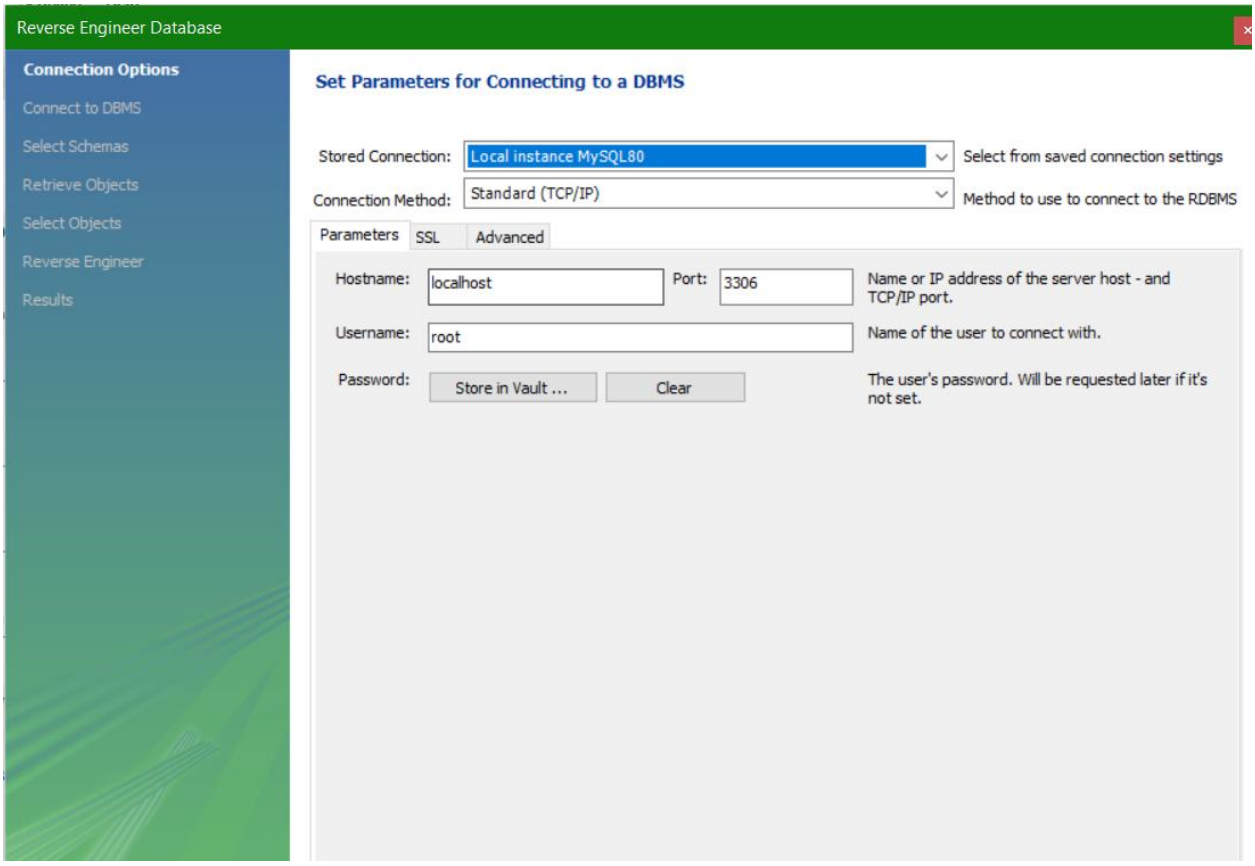


Рис.2.1 Налаштування параметрів підключення до СУБД MySQL у середовищі MySQL Workbench

Після встановлення з'єднання система автоматично виконала отримання інформації про доступні схеми бази даних та перевірила коректність

конфігурації сервера. Успішне виконання цього етапу підтвердило готовність середовища до побудови ER-моделі.

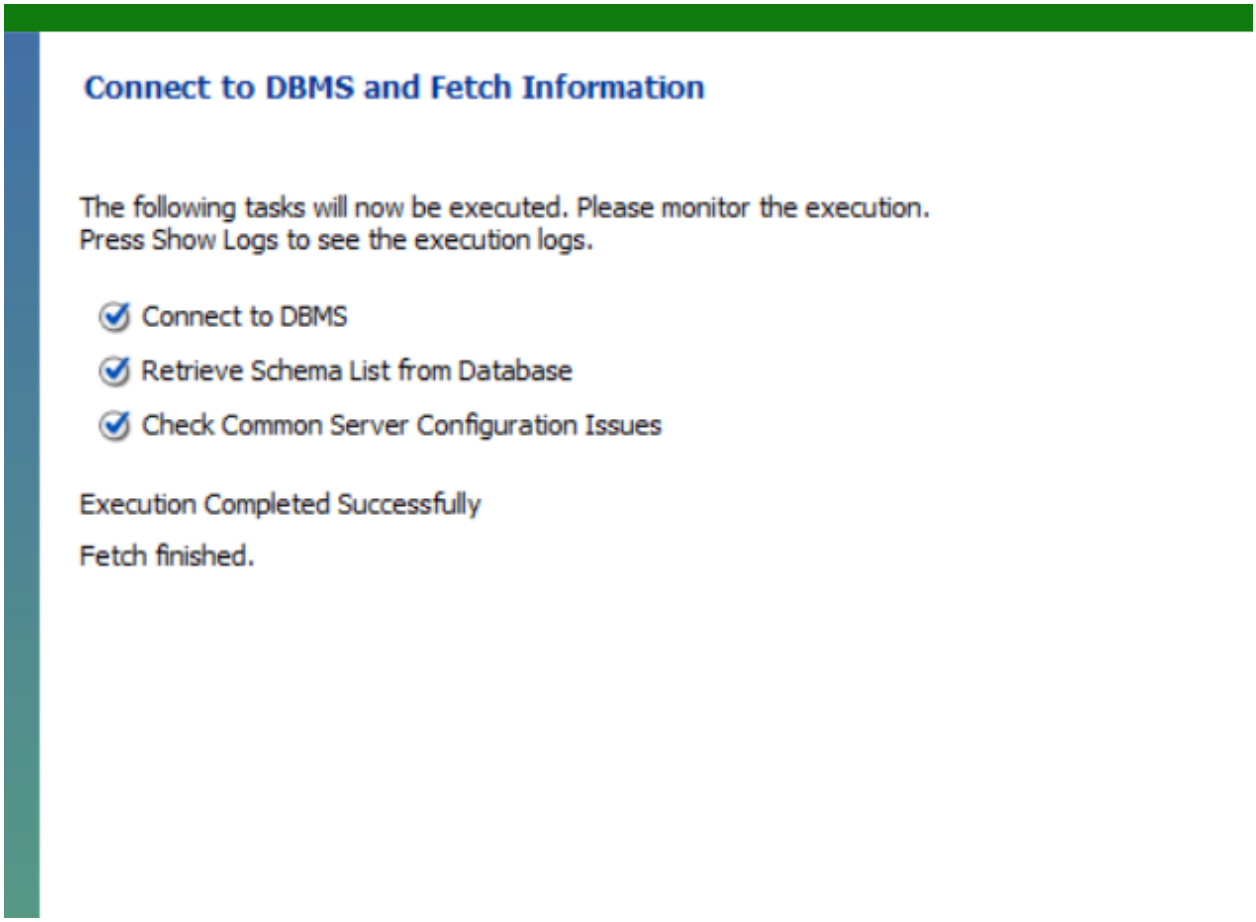


Рис. 2.2 Процес підключення до СУБД та отримання інформації про структуру бази даних

На наступному етапі було обрано схему ayvengo_analytics, яка містить таблиці доходів, витрат, користувачів та аналітичні представлення. Саме ця схема є основою функціонування аналітичної інформаційної системи.

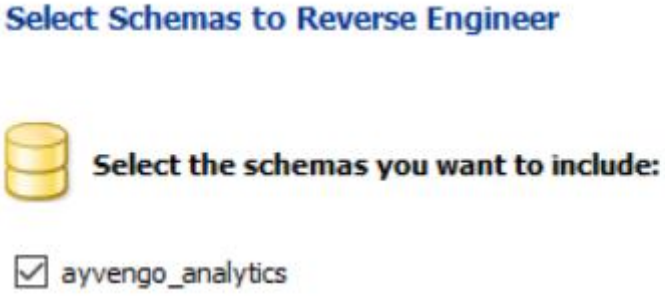


Рис. 2.3 Вибір схеми бази даних для зворотного проєктування

У результаті виконання зворотного проєктування було сформовано ER-діаграму, що відображає логічну структуру бази даних аналітичної системи.

Діаграма включає основні таблиці та аналітичні представлення, які забезпечують зберігання та обробку фінансових показників.

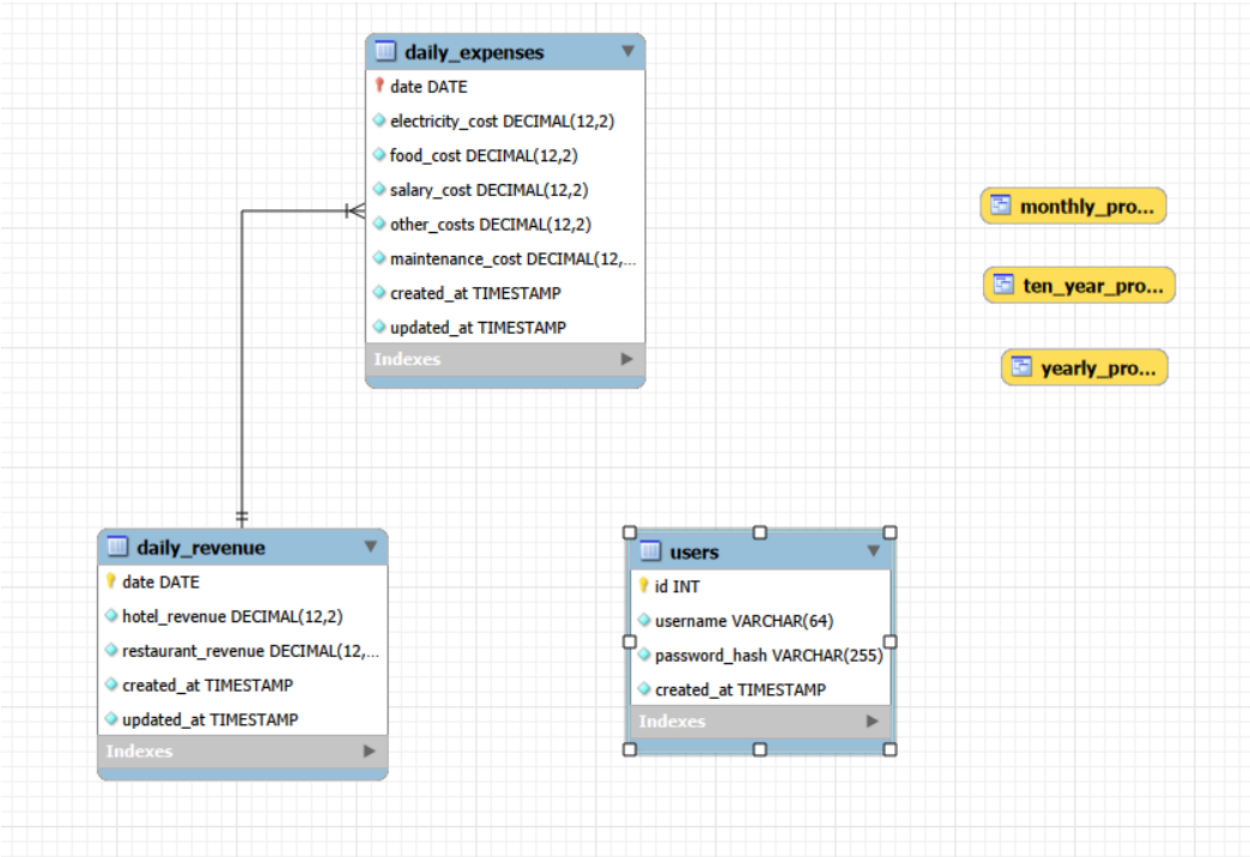


Рис. 2.4. ER-діаграма бази даних аналітичної інформаційної системи готельно-ресторанного комплексу «Айвенго»

На ER-діаграмі відображено такі ключові сутності:

- таблиця daily_revenue, що зберігає щоденні доходи готелю та ресторану;
- таблиця daily_expenses, яка містить інформацію про основні статті витрат за кожен день;
- таблиця users, призначена для автентифікації адміністратора системи;
- аналітичні представлення monthly_profit, yearly_profit та ten_year_profit, які формують агреговані показники для аналітики.

Зв’язок між daily_revenue та daily_expenses:

Тип зв’язку: 1 : 1

Ключове поле: date

Логіка:

для кожної календарної дати існує один запис доходів;

для цієї ж дати існує один запис витрат.

Це абсолютно виправдане рішення, оскільки:

- фінансова аналітика в системі будується по днях;
- всі агреговані показники (місяць, рік, 10 років) обчислюються на основі щоденних даних;
- дата є природним бізнес-ключем, а не штучним ідентифікатором.

Саме тому використання date як PRIMARY KEY в обох таблицях є коректним і доцільним.

Сутність **users** на діаграмі **свідомо не пов'язана** з таблицями доходів і витрат, адже система орієнтована на **адміністративний доступ**, а не на багатокористувацький фінансовий облік. Аналітичні представлення не включаються до фізичної ER-моделі, оскільки є логічними результатами агрегації первинних даних.

Основним зв'язком у моделі є логічний зв'язок типу «один до одного» між таблицями daily_revenue та daily_expenses, реалізований через спільний атрибут date. Такий підхід дозволяє зіставляти доходи та витрати за кожен календарний день і є оптимальним для подальшої агрегації даних.

Аналітичний режим системи реалізовано за допомогою SQL-представлень, які не зберігають дані фізично, а формують аналітичну інформацію на основі агрегування щоденних доходів і витрат. Представлення дозволяють отримувати узагальнені показники прибутку за місяць, рік та десятирічний період без дублювання даних і без необхідності додаткової обробки на рівні клієнтського застосунку.

Використання представлень у складі ER-моделі підвищує:

- цілісність аналітичних розрахунків;
- продуктивність серверної частини;
- гнучкість формування управлінських звітів.

Таким чином, ER-діаграма бази даних аналітичної системи наочно відображає логічну структуру зберігання фінансових даних готельно-ресторанного комплексу «Айвенго» та підтверджує коректність обраної моделі. Застосування зворотного проєктування забезпечило повну відповідність діаграми реальній реалізації бази даних, а використання аналітичних представлень створило ефективний механізм формування управлінської аналітики.

2.5. Проєктування архітектури web-додатку

Проєктування архітектури web-додатку аналітичної інформаційної системи готельно-ресторанного комплексу «Айвенго» здійснювалося з урахуванням сучасних підходів до побудови інформаційних систем, які орієнтовані на масштабованість, надійність та зручність супроводу. Особлива увага приділялася можливості подальшого розширення функціоналу системи без суттєвих змін у її базовій структурі.

Архітектурне рішення web-додатку ґрунтується на багаторівневій клієнт–серверній моделі, яка передбачає розподіл системи на окремі логічні рівні. Такий підхід дозволяє ізолювати обробку даних, бізнес-логіку та інтерфейс користувача, що позитивно впливає на стабільність роботи системи та спрощує її тестування і модернізацію.

Клієнтський рівень системи реалізовано у вигляді web-інтерфейсу з використанням мов HTML, CSS та JavaScript. Він забезпечує взаємодію користувача з системою, введення щоденних фінансових показників, вибір режиму роботи (скорочений або детальний), а також перегляд аналітичних звітів. Для відображення результатів аналізу використовуються таблиці та графіки, що дозволяє візуально оцінювати динаміку доходів, витрат і прибутку. Обмін даними між клієнтською частиною та сервером здійснюється за допомогою HTTP-запитів до API.

Серверний рівень web-додатку реалізований з використанням платформи Node.js та фреймворку Express. Даний рівень відповідає за обробку запитів клієнта, автентифікацію користувачів, перевірку прав доступу, а також взаємодію з базою даних. Серверна логіка забезпечує прийом і валідацію введених фінансових даних, їх збереження та подальше надання агрегованої інформації для формування аналітичних звітів. Використання Node.js дозволяє ефективно обробляти паралельні запити та забезпечувати стабільну роботу системи.

Рівень зберігання даних та аналітики реалізований на основі системи управління базами даних MySQL. База даних спроєктована з урахуванням принципів нормалізації та логічного поділу інформації. Основні таблиці містять первинні щоденні фінансові дані щодо доходів і витрат готельно-ресторанного комплексу, а також дані про користувачів системи. Для забезпечення цілісності інформації між таблицями встановлено відповідні зв’язки, зокрема за датою обліку.

Особливістю архітектури є використання SQL-представлень для реалізації аналітичного режиму системи. Представлення дозволяють формувати агреговані показники за місяць, рік та десятирічний період без дублювання даних у фізичних таблицях. Такий підхід переносить значну частину аналічної логіки на рівень бази даних, що зменшує навантаження на серверну частину та підвищує продуктивність формування звітів.

Фрагменти SQL-коду, які використовуються для створення структури бази даних, зв’язків між таблицями та аналітичних представлень, є складовою архітектурного рішення системи. Вони визначають правила зберігання інформації, механізми агрегації фінансових показників та забезпечують коректність розрахунків прибутку. Завдяки використанню зовнішніх ключів і каскадних обмежень досягається цілісність і узгодженість даних.

Аналітичний режим системи інтегрований у загальну архітектуру web-додатку та забезпечує отримання узагальненої інформації для управлінських рішень. Серверна частина взаємодіє з аналітичними представленнями бази

даних і передає клієнтському інтерфейсу вже оброблені результати. Це дозволяє спростити реалізацію візуалізації даних і забезпечити швидкий доступ до ключових фінансових показників.

Таким чином, спроєктована архітектура web-додатку забезпечує ефективну взаємодію між клієнтською частиною, серверною логікою та рівнем зберігання даних. Запропоноване архітектурне рішення дозволяє реалізувати аналітичну інформаційну систему, яка є гнучкою, масштабованою та придатною для подальшого розвитку відповідно до потреб готельно-ресторанного комплексу.

Висновки по розділу 2

У другому розділі було виконано проєктування аналітичної інформаційної системи для готельно-ресторанного комплексу «Айвенго». Сформульовано основні вимоги до майбутньої системи та визначено її функціональне призначення. Розроблено структуру даних для аналітичної обробки фінансової інформації, побудовано модель даних і ER-діаграму бази даних, що забезпечують логічну цілісність, узгодженість і можливість масштабування системи. Запроєктована архітектура створює надійну основу для подальшої програмної реалізації та формування управлінської аналітики.

РОЗДІЛ 3. РОЗРОБКА АНАЛІТИЧНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ ГРК «АЙВЕНГО»

3.1. Технологічний стек

Розробка аналітичної інформаційної системи для ГРК «Айвенго» виконувалася у форматі web-додатку з клієнт-серверною архітектурою. Серверну частину реалізовано на платформі Node.js із використанням фреймворку Express, що забезпечує обробку HTTP-запитів, маршрутизацію та організацію API. Для зберігання первинних даних і виконання аналітичних агрегувань застосовано MySQL, а для адміністрування та проєктування структури – MySQL Workbench.

Розроблена система для ГРК «Айвенго» реалізована як web-додаток із розділенням на серверну та клієнтську частини. Основна ідея стеку полягає в тому, щоб забезпечити: надійне збереження фінансових даних, швидке формування агрегованих показників, зручну візуалізацію та порівняння періодів у браузері.

Серверна частина (Back-end):

1. Node.js – середовище виконання JavaScript на сервері, зручне для побудови API та роботи з введенням даних у форматі JSON.
2. Express – мінімалістичний web-фреймворк для маршрутизації, middleware та реалізації REST-ендпоінтів.
3. bcryptjs – хешування/перевірка паролів адміністратора (захист доступу).
4. cookie-parser – робота з cookie для авторизації.
5. dotenv – зберігання конфігурації (порт, секрет сесії, параметри БД) у .env.

Робота з даними (Data layer):

1. MySQL – реляційна СКБД для збереження щоденних доходів/витрат.
2. SQL VIEW – для формування аналітики без дублювання даних.

3. mysql2/promise – пул з’єднань, транзакції, параметризовані запити.

Клієнтська частина (Front-end):

HTML + CSS + JavaScript – статичний інтерфейс.

Chart.js – побудова графіків (динамічна візуалізація доходів/витрат/прибутку, а також порівняння періодів).

Основні сторінки:

1. login.html – форма входу адміністратора.
2. admin.html – меню (перехід до введення даних і до звітів).
3. entry.html – введення щоденних показників за обраний місяць.
4. report.html – аналітика: місяць/рік/10 років + порівняння періодів.

Інструменти розробки:

1. Visual Studio Code – написання коду.
2. MySQL Workbench – створення структури БД, reverse engineering та ER-діаграма.

Для підключення до MySQL у серверній частині використано бібліотеку mysql2, що дає змогу працювати з базою даних через пул підключень та виконувати транзакції. Налаштування середовища і параметрів підключення здійснено за допомогою dotenv. Для реалізації авторизації адміністратора використано cookie-parser та bcryptjs для перевірки хешу пароля.

3.2. Створення та налаштування бази даних

База даних ayvengo_analytics створювалась як реляційна схема для зберігання щоденних фінансових показників і подальшого формування узагальнених звітів. На етапі ініціалізації встановлюється кодування utf8mb4, що дозволяє коректно працювати з українськими назвами й текстовими полями. Далі створюється база даних і виконується очищення об’єктів – для «чистого старту»(видалення представлень та таблиць).

У структурі схеми використано три ключові таблиці: users, daily_revenue та daily_expenses. Таблиця users призначена для автентифікації адміністратора і містить логін та bcrypt-хеш пароля. Таблиця daily_revenue зберігає щоденні доходи за двома напрямами (готель і ресторан) з ключем за датою. Таблиця daily_expenses акумулює витрати за основними статтями також з ключем за датою.

Важливим архітектурним рішенням є те, що daily_expenses має зовнішній ключ на daily_revenue(date) із каскадними правилами оновлення та видалення. Це дисциплінує дані: витрати логічно прив’язані до конкретного дня обліку та не «висять» без відповідної дати. Додатково створено індекси за датою, що пришвидшує фільтрацію та агрегацію на часових проміжках.

Аналітична частина реалізована через SQL-представлення: monthly_profit, yearly_profit, ten_year_profit. Представлення формують узагальнені показники без дублювання даних у фізичних таблицях. Зокрема, monthly_profit агрегує сумарний дохід і сумарні витрати по місяцях, а прибуток обчислюється як різниця між ними. Аналогічно yearly_profit виконує агрегацію по роках, а ten_year_profit надає «вітрину»за останні 10 років, що використовується для довгострокового аналізу.

Ініціалізація БД та кодування:

```
SET NAMES utf8mb4;
CREATE DATABASE IF NOT EXISTS ayvengo_analytics
 CHARACTER SET utf8mb4
 COLLATE utf8mb4_unicode_ci;
USE ayvengo_analytics;
```

Початковий блок забезпечує коректне кодування та створює схему ayvengo_analytics.

Скидання об’єктів для коректної роботи:

```
DROP VIEW IF EXISTS ten_year_profit;
DROP VIEW IF EXISTS yearly_profit;
DROP VIEW IF EXISTS monthly_profit;
```

```
DROP TABLE IF EXISTS daily_expenses;
DROP TABLE IF EXISTS daily_revenue;
DROP TABLE IF EXISTS users;
```

Даний фрагмент дозволяє повторно запускати скрипт без помилок. Спершу видаляються VIEW, оскільки вони залежать від таблиць, а потім – таблиці.

Таблиця користувачів:

```
CREATE TABLE users (
id INT UNSIGNED NOT NULL AUTO_INCREMENT,
username VARCHAR(64) NOT NULL,
password_hash VARCHAR(255) NOT NULL,
created_at TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
PRIMARY KEY (id),
UNIQUE KEY uq_users_username (username)
) ENGINE=InnoDB;
```

Таблиця users зберігає облікові записи адміністраторів. Замість пароля зберігається **хеш** (password_hash), що відповідає базовим вимогам безпеки. Унікальний ключ на username запобігає дублюванню логінів.

Таблиця щоденних доходів:

```
CREATE TABLE daily_revenue (
date DATE NOT NULL,
hotel_revenue DECIMAL(12,2) NOT NULL DEFAULT 0.00,
restaurant_revenue DECIMAL(12,2) NOT NULL DEFAULT 0.00,
created_at TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
updated_at TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
PRIMARY KEY (date),
```

```
  KEY idx_daily_revenue_date (date)
) ENGINE=InnoDB;
```

daily_revenue фіксує доходи за день окремо для готелю і ресторану. Первинний ключ date гарантує один запис на дату, а індекс по даті прискорює вибірки для звітів.

Таблиця щоденних витрат і контроль цілісності:

```
CREATE TABLE daily_expenses (
  date DATE NOT NULL,
  electricity_cost DECIMAL(12,2) NOT NULL DEFAULT 0.00,
  food_cost        DECIMAL(12,2) NOT NULL DEFAULT 0.00,
  salary_cost      DECIMAL(12,2) NOT NULL DEFAULT 0.00,
  other_costs      DECIMAL(12,2) NOT NULL DEFAULT 0.00,
  maintenance_cost DECIMAL(12,2) NOT NULL DEFAULT 0.00,
  created_at TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
  updated_at TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  PRIMARY KEY (date),
  KEY idx_daily_expenses_date (date),
  CONSTRAINT fk_expenses_to_revenue
    FOREIGN KEY (date) REFERENCES daily_revenue(date)
    ON DELETE CASCADE
    ON UPDATE CASCADE
) ENGINE=InnoDB;
```

Таблиця витрат логічно прив’язана до доходів того ж дня через зовнішній ключ fk_expenses_to_revenue. Це унеможливлює «висячі» витрати без відповідного дня в доходах та забезпечує узгодженість при видаленні/оновленні дати.

Аналітичні представлення (VIEW):

```
//Місячний прибуток
```

```
CREATE VIEW monthly_profit AS
SELECT
 STR_TO_DATE(DATE_FORMAT(dr.date, '%Y-%m-01'), '%Y-%m-%d')
AS month,
 CAST(SUM(dr.hotel_revenue + dr.restaurant_revenue) AS
DECIMAL(14,2)) AS total_revenue,
 CAST(SUM(
   COALESCE(de.electricity_cost,0) +
   COALESCE(de.food_cost,0) +
   COALESCE(de.salary_cost,0) +
   COALESCE(de.other_costs,0) +
   COALESCE(de.maintenance_cost,0)
 ) AS DECIMAL(14,2)) AS total_expenses,
 CAST(
  SUM(dr.hotel_revenue + dr.restaurant_revenue) -
  SUM(
   COALESCE(de.electricity_cost,0) +
   COALESCE(de.food_cost,0) +
   COALESCE(de.salary_cost,0) +
   COALESCE(de.other_costs,0) +
   COALESCE(de.maintenance_cost,0)
  )
 AS DECIMAL(14,2)) AS profit
FROM daily_revenue dr
LEFT JOIN daily_expenses de ON de.date = dr.date
GROUP BY STR_TO_DATE(DATE_FORMAT(dr.date, '%Y-%m-01'),
'%Y-%m-%d')
ORDER BY month;
```

Це представлення агрегує дані за місяць: дохід, витрати, прибуток. COALESCE використовується для підстановки нулів у випадку відсутніх витрат, щоб формули працювали стабільно.

Річний та десятирічний прибуток:

```
CREATE VIEW yearly_profit AS
SELECT
  STR_TO_DATE(CONCAT(YEAR(dr.date), '-01-01'), '%Y-%m-%d') AS year_start,
  CAST(SUM(dr.hotel_revenue + dr.restaurant_revenue) AS DECIMAL(14,2)) AS total_revenue,
  CAST(SUM(
    COALESCE(de.electricity_cost,0) +
    COALESCE(de.food_cost,0) +
    COALESCE(de.salary_cost,0) +
    COALESCE(de.other_costs,0) +
    COALESCE(de.maintenance_cost,0)
  ) AS DECIMAL(14,2)) AS total_expenses,
  CAST(
   SUM(dr.hotel_revenue + dr.restaurant_revenue) -
   SUM(
    COALESCE(de.electricity_cost,0) +
    COALESCE(de.food_cost,0) +
    COALESCE(de.salary_cost,0) +
    COALESCE(de.other_costs,0) +
    COALESCE(de.maintenance_cost,0)
   )
  AS DECIMAL(14,2)) AS profit
FROM daily_revenue dr
LEFT JOIN daily_expenses de ON de.date = dr.date
GROUP BY YEAR(dr.date)
```

```
ORDER BY year_start;
CREATE VIEW ten_year_profit AS
SELECT *
FROM yearly_profit
WHERE YEAR(year_start) >= YEAR(CURDATE()) - 9
ORDER BY year_start;
```

Дані VIEW використовуються фронтендом для режимів «Рік» та «10 років». Ідея полягає у тому, щоб на рівні БД одразу сформувати потрібні агрегати, а не виконувати підрахунки на клієнті.

Додавання адміністратора та перевірка:

```
INSERT INTO users (username, password_hash)
VALUES ('admin', '$2b$10$6lK5t4ppHGoeO4.cIqyfX.XelsCwn5Lj4g6NTm7R/7Gao9nE4v4qe')
ON DUPLICATE KEY UPDATE password_hash = VALUES(password_hash);
SELECT * FROM monthly_profit ORDER BY month LIMIT 12;
SELECT * FROM yearly_profit ORDER BY year_start LIMIT 12;
SELECT * FROM ten_year_profit ORDER BY year_start LIMIT 12;
```

Створюється тестовий адмін-акаунт та виконуються контрольні запити. Що це дає, нам це дозволяє одразу переконатися, що структура готова і представлення повертають коректні поля. Усі ключові моменти враховано та виконано для вдалого запуску проєкту.

Після створення структури додається запис адміністратора з попередньо згенерованим bcrypt-хешем пароля. Останнім кроком можна вважати є виконання контрольних запитів (підрахунок рядків у таблицях та вибірка з представлень), що дозволяє оперативно переконатися в коректності створення бази та готовності до роботи сервера.

3.3. Серверна логіка: імпорт та збереження щоденної виручки

Серверна логіка web-додатку реалізована у вигляді набору маршрутів Express. Доступ до функціоналу введення даних та перегляду звітів обмежений авторизацією адміністратора. Після успішного входу сервер встановлює підписаний cookie-прапорець, який надалі використовується в проміжному обробнику перевірки доступу. Такий підхід дозволяє відокремити відкриті сторінки (форма входу) від захищених (введення даних, аналітика).

Імпорт щоденних показників виконано через HTTP-запит з клієнтської частини до API, який передає масив рядків таблиці. На сервері ці дані обробляються в транзакції: для кожного дня виконується запис/оновлення (UPSERT) доходів у daily_revenue і, залежно від режиму, запис/оновлення витрат у daily_expenses. Якщо режим скорочений, витрати приводяться до нульових значень, що робить модель даних однорідною для подальших агрегувань.

Серверна частина реалізує:

- автентифікацію адміністратора,
- прийом щоденних даних з форми введення,
- збереження в БД у транзакції,
- видачу аналітики у форматі JSON для графіків.

Авторизація адміністратора:

```
app.post('/login', async (req, res) => {
  const { username, password } = req.body;
  const [rows] = await pool.query(
    'SELECT id, username, password_hash FROM users WHERE username = ? LIMIT 1',
    [username]
  );
```

```
  const ok = await bcrypt.compare(password, rows[0].password_hash);
  if (!ok) return res.status(401).send('Невірний логін або пароль');
  res.cookie('isAdmin', '1', { httpOnly: true, signed: true, sameSite: 'lax', maxAge: 8*60*60*1000 });
  return res.redirect('/admin.html');
});
```

Логіка входу зводиться до перевірки хешу пароля через bcrypt.compare(). Успішний вхід встановлює signed-cookie, яке далі використовується middleware-перевіркою доступу.

```
function requireAuth(req, res, next) {
  if (req.signedCookies && req.signedCookies.isAdmin === '1') return next();
  return res.redirect('/login');
}
```

Це простий, але надійний спосіб обмежити доступ до сторінок і API, не ускладнюючи систему важкими механізмами сесій.

Прийом та збереження щоденних записів:

```
app.post('/api/daily-data', requireAuth, async (req, res) => {
 const { rows, mode } = req.body;
 conn = await pool.getConnection();
 await conn.beginTransaction();
  for (const r of rows) {
    const date = r.date;
    await conn.query(
      `INSERT INTO daily_revenue (date, hotel_revenue, restaurant_revenue)
       VALUES (?, ?, ?)
       ON DUPLICATE KEY UPDATE
         hotel_revenue = VALUES(hotel_revenue),
         restaurant_revenue = VALUES(restaurant_revenue)`,
      [date, Number(r.hotel_revenue || 0), Number(r.restaurant_revenue || 0)]
    );
```

```
        if (mode === 'full') {
          await conn.query(
            `INSERT INTO daily_expenses
             (date, electricity_cost, food_cost, salary_cost, other_costs,
maintenance_cost)
             VALUES (?, ?, ?, ?, ?, ?)
             ON DUPLICATE KEY UPDATE
               electricity_cost = VALUES(electricity_cost),
               food_cost = VALUES(food_cost),
               salary_cost = VALUES(salary_cost),
               other_costs = VALUES(other_costs),
               maintenance_cost = VALUES(maintenance_cost)`,
            [date, Number(r.electricity_cost||0), Number(r.food_cost||0),
Number(r.salary_cost||0), Number(r.other_costs||0),
Number(r.maintenance_cost||0)]
          );
        }
      }
      await conn.commit();
      res.json({ ok: true });
    });
```

Якщо хоча б один рядок не запишеться, система робить rollback, щоб уникнути частково збережених періодів. ON DUPLICATE KEY UPDATE дозволяє коригувати дані без створення дублюючих записів.

Отримання аналітики для графіків;

Сервер повертає агреговані дані з VIEW:

```
app.get('/api/monthly-profit', requireAuth, async (req, res) => {
  const [rows] = await pool.query(
```

```
    'SELECT month, total_revenue, total_expenses, profit FROM
monthly_profit ORDER BY month'
    );
    res.json(rows);
});
```

У результаті фронтенд не рахує фінанси самостійно – він лише відображає те, що вже підготувала БД.Контрольні запити використовуються для перевірки коректності розрахунків і структури результатів, що надалі передаються через API до клієнтської частини системи.

У даному підрозділі наведено приклади інтерфейсу та результатів роботи розробленої аналітичної web-системи для ГРК «Айвенго». Представлені зображення демонструють основні функціональні можливості системи, а також оновлений візуальний стиль користувацького інтерфейсу.

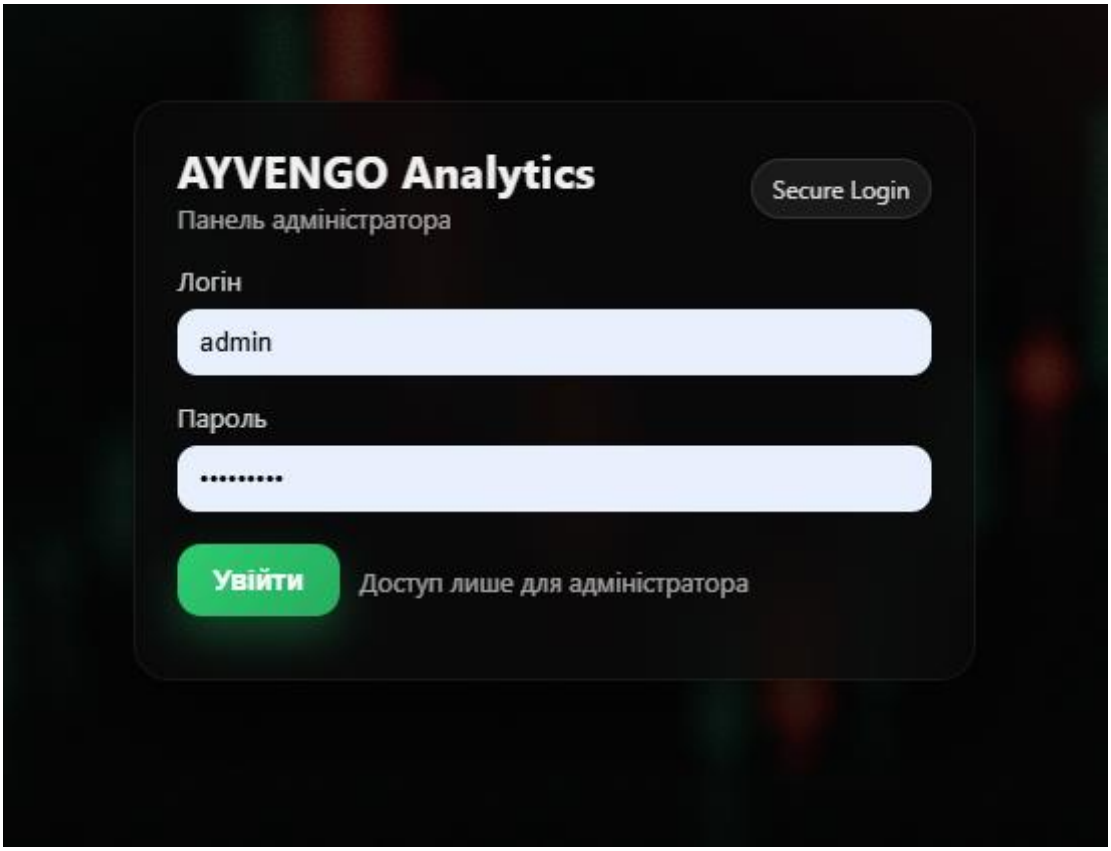


Рис. 3.1 Сторінка авторизації адміністратора

На рисунку зображено сторінку входу до системи **AYVENGO Analytics**, яка призначена виключно для адміністратора. Інтерфейс містить поля для

введення логіна та пароля, а також кнопку підтвердження входу. Доступ до системи обмежений та реалізований із використанням хешування паролів у базі даних, що підвищує рівень інформаційної безпеки. Візуальне оформлення виконано у темній кольоровій гамі з акцентами зеленого кольору, що підкреслює аналітичний характер системи.

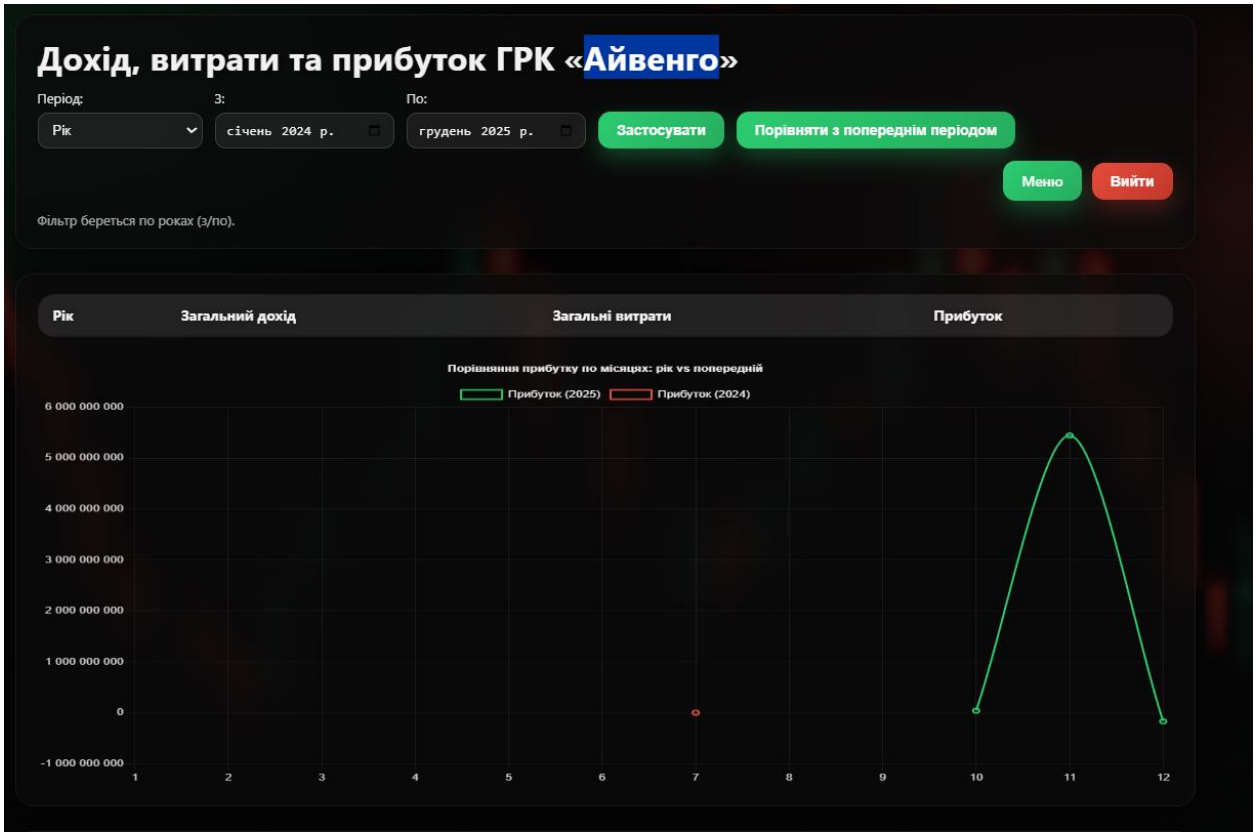


Рис. 3.2 Головна сторінка аналітики

Даний екран відображає головну сторінку аналітичної панелі з обраним періодом аналізу **«Рік»**. Користувач має змогу задати часові межі аналізу, застосувати фільтри та порівняти фінансові показники з попереднім періодом. Нижче представлено графік динаміки прибутку по місяцях для різних років, що дозволяє виявляти тенденції зростання або спаду фінансових результатів.

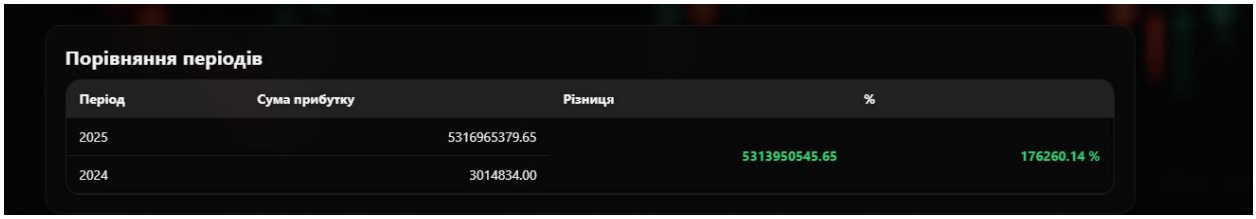


Рис. 3.3 Таблиця порівняння періодів

На рисунку наведено таблицю порівняння фінансових результатів за два роки. Таблиця містить такі показники:

- загальну суму прибутку;
- абсолютну різницю між періодами;
- відносну зміну у відсотках.

Застосування табличного представлення дозволяє швидко оцінити ефективність діяльності комплексу в динаміці.

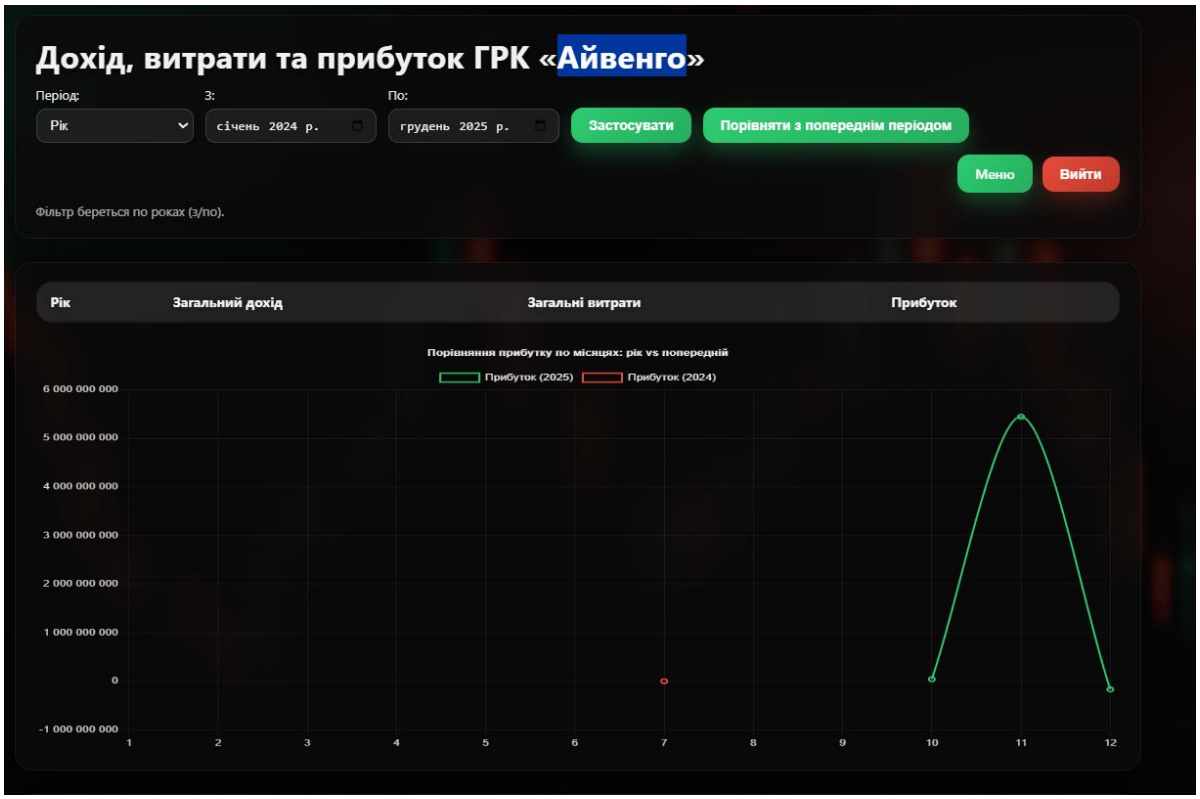


Рис. 3.4 Аналітика за місяцями з використанням фільтрів

Даний інтерфейс демонструє роботу системи у режимі місячного аналізу. Користувач може обрати період «з/по», застосувати фільтр та отримати агреговані фінансові показники. Такий підхід дозволяє проводити детальний аналіз короткострокових змін у доходах і витратах.

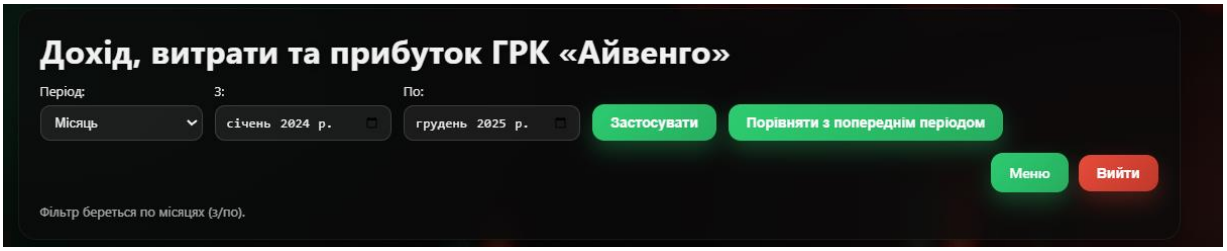


Рис. 3.5 Форма введення щоденних фінансових даних

На рисунку 3.6 показано екран введення щоденних даних, який використовується для наповнення бази даних. Форма містить поля для:

- доходів готелю;
- доходів ресторану;
- витрат на електроенергію;
- витрат на продукти;
- заробітну плату;
- інші та обслуговуючі витрати.

Передбачено режими детального та скороченого введення даних, що спрощує роботу користувача.

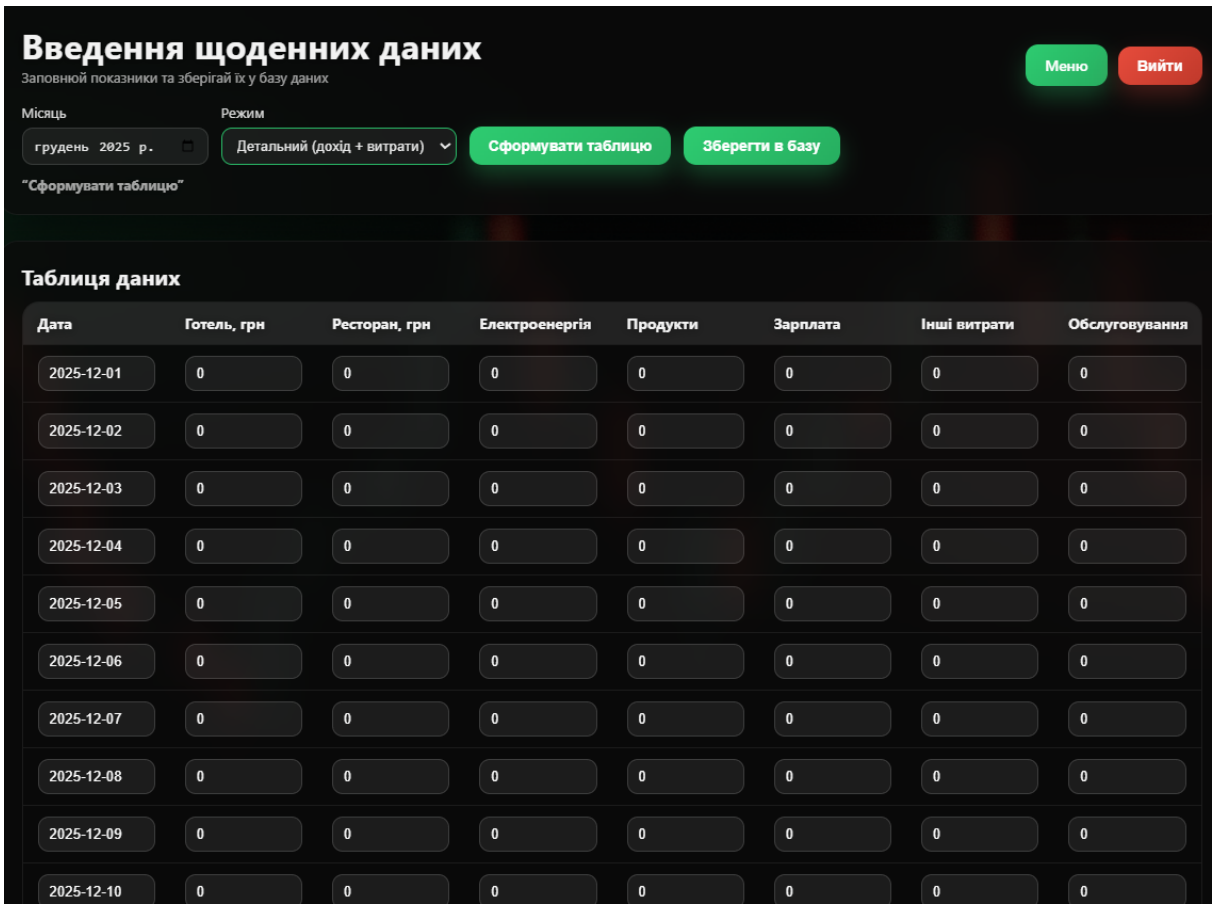


Рис. 3.6 Таблиця та графік доходів, витрат і прибутку за місяцями

На рисунку 3.7 представлено комбіноване відображення результатів аналізу:

- таблична частина з підсумковими значеннями;

- графічна візуалізація доходів, витрат та прибутку.

Завдяки використанню кольорового розмежування інформація легко сприймається та швидко аналізується.

Рис. 3.7 Порівняння прибутку між двома місяцями

На даному рисунку зображено графік порівняння прибутку між поточним та попереднім місяцем. Такий інструмент дозволяє оперативно виявляти аномальні значення, різкі коливання або від'ємний показник.

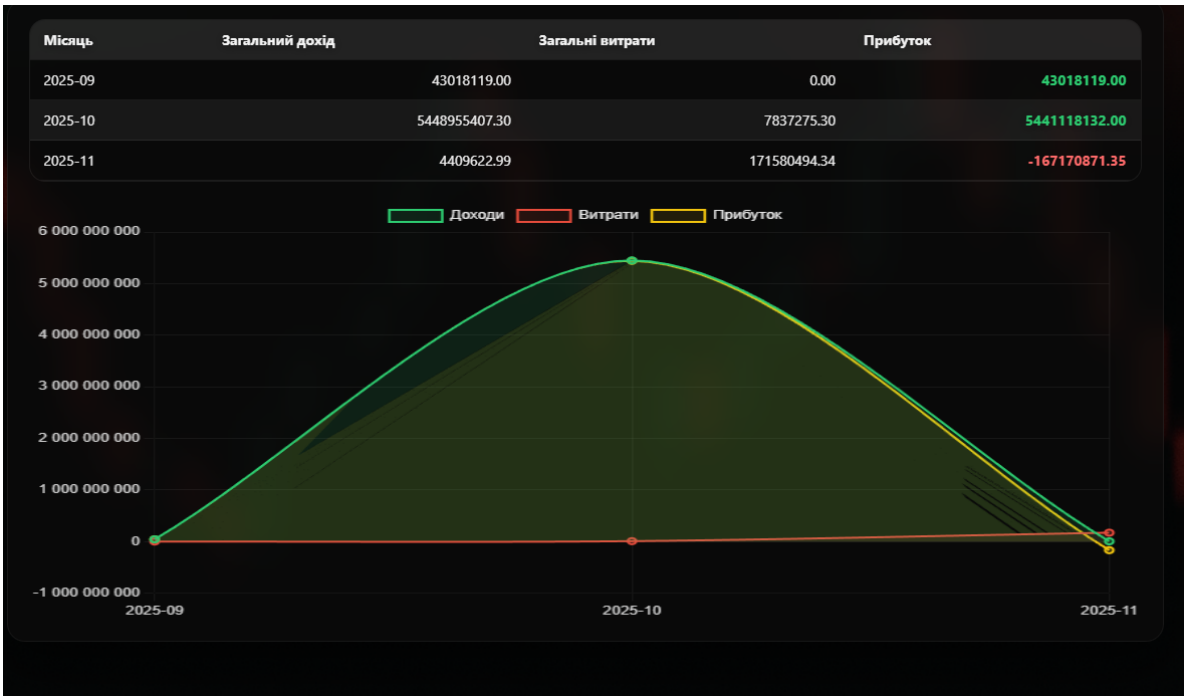


Рис. 3.8 Оновлений стиль інтерфейсу аналітичної системи

Даний екран демонструє оновлений дизайн системи з використанням темної теми, розмитого фонового зображення та сучасних UI-елементів. Такий підхід підвищує зручність роботи з аналітичними даними, зменшує навантаження на зір та покращує загальне сприйняття інформації.

Представлені скриншоти підтверджують працездатність розробленої інформаційної системи та її відповідність поставленим вимогам. Реалізований інтерфейс забезпечує зручний доступ до аналітичних даних, підтримує порівняльний аналіз і візуалізацію фінансових показників, що робить систему ефективним інструментом для управління діяльністю.

Застосування транзакційного підходу є принциповим або зберігається весь набір даних за період, або у випадку помилки зміни відкатуються. Це забезпечує узгодженість даних і коректність звітів. На рівні сервера також реалізовано обробку винятків і повернення структурованих відповідей клієнту (успіх/помилка), що спрощує тестування та діагностику.

Для формування звітів реалізовано API-маршрути, які звертаються до аналітичних представлень у MySQL. Наприклад, GET /api/monthly-profit повертає агрегацію по місяцях. Аналогічні маршрути можуть повертати деталізацію витрат або інші вітрини аналітики.

3.4. Реалізація інтерфейсу web-додатку

Інтерфейс web-додатку спроєктовано з урахуванням типових робочих сценаріїв адміністратора готельно-ресторанного комплексу та орієнтацією на швидке виконання щоденних операцій. Логіка взаємодії побудована таким чином, щоб користувач послідовно проходив усі етапи роботи із системою без зайвих переходів і дублювання дій. Початковою точкою доступу є сторінка автентифікації, яка забезпечує контроль доступу до аналітичної інформації та запобігає несанкціонованому використанню системи.

Після успішного входу користувач потрапляє до адміністративного меню, яке виконує роль центрального навігаційного елемента. Звідси здійснюється перехід до сторінки введення щоденних фінансових даних або до модуля аналітичних звітів. Така структура інтерфейсу відповідає принципу функціонального розмежування та спрощує орієнтацію в системі навіть для користувачів без технічної підготовки.

Сторінка введення даних реалізує адаптивний підхід до формування форм. Користувач обирає календарний місяць і режим заповнення — скорочений або детальний, після чого система автоматично генерує таблицю з кількістю рядків, що відповідає числу днів у вибраному періоді. Це рішення зменшує кількість ручних операцій, виключає помилки у виборі дат та забезпечує уніфікований формат введення інформації. Після заповнення таблиці дані передаються на сервер у форматі JSON, що забезпечує зручну інтеграцію з серверною логікою та базою даних.

Інтерфейс клієнтської частини реалізовано у вигляді набору статичних HTML-сторінок із використанням CSS та JavaScript, які обслуговуються сервером через механізм express.static. Такий підхід дозволив мінімізувати складність розгортання та зосередити серверну частину виключно на обробці даних і бізнес-логіці. Обмін інформацією між клієнтом і сервером здійснюється через REST-ендпоінти, що відповідає сучасним підходам до побудови web-додатків.

Особлива увага приділена узгодженості візуального стилю та навігації. Єдина кольорова схема, повторювані елементи керування («Меню», «Вийти»), а також чітке розташування основних блоків інтерфейсу сприяють зручності використання та зменшують когнітивне навантаження на користувача. Інтерфейс не перевантажений другорядними елементами, що відповідає вимогам до прикладних управлінських систем.

Сторінка аналітичних звітів забезпечує комплексне представлення фінансової інформації за обраний період. Дані, отримані з серверного API, одночасно відображаються у табличній формі та у вигляді графіків, що

дозволяє поєднати точність числових значень із наочністю візуального аналізу. Такий підхід підвищує ефективність роботи з аналікою та дозволяє оперативно виявляти тенденції, відхилення та динаміку ключових показників діяльності підприємства.

3.5. Візуалізація даних та формування управлінських звітів

У розробленій системі візуалізація фінансових показників реалізована на основі бібліотеки **Chart.js**, що дає змогу представити числові дані у наочній формі та підвищити швидкість їх інтерпретації. На відміну від статичних таблиць, графічне подання дозволяє оперативно оцінити динаміку виручки, витрат і прибутку, помітити нетипові коливання, а також порівняти результати різних періодів без додаткових ручних розрахунків.

Побудова звітів у системі орієнтована на типові управлінські запити, які виникають у діяльності готельно-ресторанного комплексу. Зокрема, система дає можливість швидко відповісти на практичні питання:

- який прибуток отримано за вибраний місяць або рік;
- як змінювалися фінансові результати протягом періоду;
- наскільки поточні показники відрізняються від попереднього аналогічного періоду.

Таким чином, візуалізація виконує не декоративну, а прикладну функцію: підтримує аналіз тенденцій, оцінку ефективності та первинну діагностику фінансових відхилень.

Клієнтська частина отримує дані для побудови графіків у форматі **JSON** через REST-ендпоінти серверної частини. Для формування помісячної аналітики використовується маршрут:

http://localhost:3000/api/monthly-profit

У разі потреби деталізації витрат може застосовуватися додатковий маршрут:

http://localhost:3000/api/monthly-expenses

Принципово важливим є те, що структура відповіді API відповідає полям SQL-представлень. Завдяки цьому фронтенд не виконує складних перетворень або обчислень, а працює з підготовленими агрегованими показниками. Такий підхід зменшує ризик логічних помилок у розрахунках і забезпечує єдині правила агрегації для всіх сценаріїв перегляду звітів.

Підключення Chart.js здійснюється через CDN:

<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>

Після отримання відповіді від API дані перетворюються у масиви, які є стандартними для побудови графіка: labels (підписи осі часу), revenue, expenses, profit. На практиці це дозволяє реалізувати інтерактивне відображення: легенду, підказки при наведенні курсора та динамічну зміну графіка при зміні фільтрів.

Окремим функціональним елементом системи є режим порівняння з попереднім періодом. У цьому режимі на одному графіку відображаються дві серії даних прибутку: поточний період та попередній. Це дає змогу оцінювати не тільки абсолютні значення, а й характер змін: чи зростання прибутку є стабільним, чи має випадкові піки, а також у які саме дні (або місяці) виникло найбільше відхилення.

Для підсилення управлінської корисності порівняння доповнюється табличним блоком, у якому відображаються підсумки: сума, різниця та відсоткова зміна. Такий формат зручний для представлення результатів керівництву, оскільки поєднує наочність графіка з точністю числових оцінок.

Формування звітів у системі базується на даних, отриманих із SQL-представлень (VIEW), які реалізують агрегацію доходів і витрат у потрібних часових розрізах. Це рішення забезпечує, що всі розрахунки виконуються на рівні бази даних за єдиними правилами, а отже не залежать від браузера, пристрою або особливостей клієнтського середовища.

У процесі розробки та тестування коректність відповіді перевірялася не лише через інтерфейс, але й шляхом прямого звернення до ендпоінтів у

браузері. Наприклад, перехід за адресою http://localhost:3000/api/monthly-profit дозволяє швидко отримати JSON-відповідь і переконатися, що структура даних та значення сформовані коректно, навіть без завантаження сторінки звітів. Така перевірка є практично зручною під час налагодження системи та дозволяє швидко локалізувати помилки між рівнем бази даних, серверною логікою та клієнтським відображенням.

3.6. Тестування працездатності системи

Перевірка працездатності розробленої аналітичної інформаційної системи здійснювалась у форматі функціонального тестування, орієнтованого на реальні сценарії роботи адміністратора готельно-ресторанного комплексу. Основна увага приділялася перевірці коректності взаємодії між клієнтською частиною, серверною логікою та базою даних, а також стабільності роботи системи під час повторних операцій.

Тестування проводилось поетапно, відповідно до логіки використання системи, починаючи з процесу авторизації та завершуючи аналізом сформованих аналітичних звітів.

На першому етапі перевірялася система автентифікації та контролю доступу**.** Було протестовано сценарії входу з коректними та некоректними обліковими даними. У разі правильного введення логіна і пароля система надавала доступ до адміністративного меню, тоді як введення помилкових даних супроводжувалося відповідним повідомленням про помилку.

Окремо перевірявся захист закритих сторінок: спроба прямого відкриття сторінки адміністрування без встановленого cookie призводила до автоматичного перенаправлення користувача на сторінку входу.

Другий етап тестування був присвячений збереженню та оновленню фінансових даних**.** Перевірялося введення щоденних показників за обраний місяць та їх коректний запис у таблиці daily_revenue і daily_expenses. Особливу

увагу приділено перевірці повторного збереження даних за той самий календарний день. У цьому випадку система повинна була не створювати дублікати, а оновлювати існуючі записи, що підтвердило коректність реалізації механізму ON DUPLICATE KEY UPDATE (UPSERT-логіки).

На наступному етапі виконувалась перевірка аналітичних модулів. Контролювалась наявність агрегованих даних у SQL-представленнях monthly_profit, yearly_profit та ten_year_profit. Значення доходів, витрат і прибутку аналізувалися на предмет логічної узгодженості, зокрема перевірялась відповідність розрахунку «дохід – витрати = прибуток». Також тестувалась коректність агрегації даних за кілька місяців та років.

Окремий етап був **присвячений** перевірці інтерфейсу користувача. Оцінювалася правильність роботи фільтрів вибору періоду («місяць», «рік», «10 років»), коректність перемикання режимів відображення та відповідність графіків обраним параметрам. Додатково перевірялось візуальне відображення від’ємних значень прибутку, які підсвічуються як індикатор збитковості, що підвищує наочність аналітики.

Особливу увагу було приділено крайовим і нетиповим випадкам**.** Зокрема, тестувалась робота системи з місяцями різної тривалості, ситуації з нульовими витратами у скороченому режимі введення даних, а також коректність відображення від’ємних фінансових результатів. Усі зазначені сценарії оброблялися системою без помилок, що підтвердило стабільність і надійність реалізованого рішення.

Результати тестування засвідчили, що розроблена аналітична інформаційна система коректно виконує всі передбачені функції, забезпечує цілісність даних і може бути використана в практичній діяльності готельно-ресторанного комплексу для підтримки управлінських рішень.

3.7. Скриншоти роботи інформаційної системи

У цьому підрозділі наведені скриншоти, які фіксують ключові сценарії використання системи та підтверджують коректність її роботи «від даних до звіту»: отримання агрегованих показників через API, виконання авторизації адміністратора, введення щоденних значень і формування управлінської аналітики у вигляді таблиць та графіків. Оскільки інтерфейс і серверна частина пов’язані спільними правилами агрегації (SQL-представленнями), скриншоти демонструють узгодженість даних на всіх рівнях: база даних, API , браузер.

Рис. 3.9 ілюструє приклад відповіді API з агрегованими фінансовими показниками по місяцях у форматі JSON (джерело – представлення monthly_profit). У відповіді присутні поля періоду та підсумкових значень: загальний дохід, загальні витрати і прибуток. Цей формат використовується клієнтською частиною без додаткових складних перетворень: отримані поля напряму переносяться у таблицю звіту та масиви для побудови графіка.

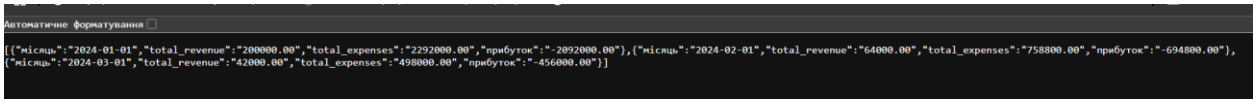


Рис. 3.9 Приклад відповіді API з агрегованими показниками прибутку

Наступний рисунок демонструє приклад відповіді API зі структурою витрат у розрізі місяця. На практиці такий тип відповіді потрібен для глибшого аналізу: дозволяє не лише бачити загальну суму витрат, а й розкривати її складники. Це важливо для управлінських рішень, коли причина зміни прибутку пов’язана не з виручкою, а з конкретною статтею витрат.

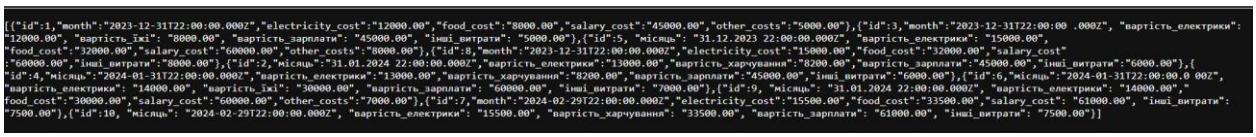


Рис. 3.10 Приклад відповіді API зі структурою витрат у розрізі місяців

Далі Рис. 3.11 містить приклад сторінки авторизації адміністратора на ранньому етапі. Вона демонструє мінімально необхідний функціонал входу: поля логіну й пароля та кнопку підтвердження. Цей скриншот доцільно

розглядати як контрольну «точку старту», від якої далі виконувалось вдосконалення UX/UI без зміни логіки безпеки.

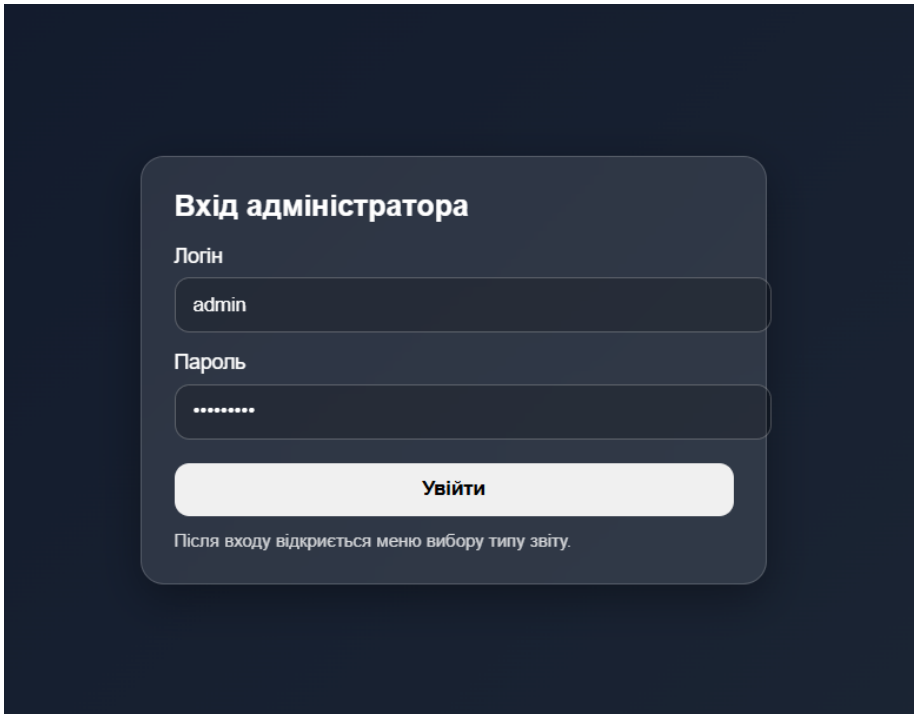


Рис. 3.11 Не стилізована сторінка авторизації адміністратора системи

Далі рисунок відображає інтерфейс введення щоденних даних у скороченому режимі. На екрані показано таблицю з датами та полями для внесення доходів (готель/ресторан). Важливим елементом є механізм автогенерації рядків після вибору місяця: система формує перелік дат відповідно до календаря, що зменшує ризик помилок при ручному введенні та прискорює заповнення первинних даних.

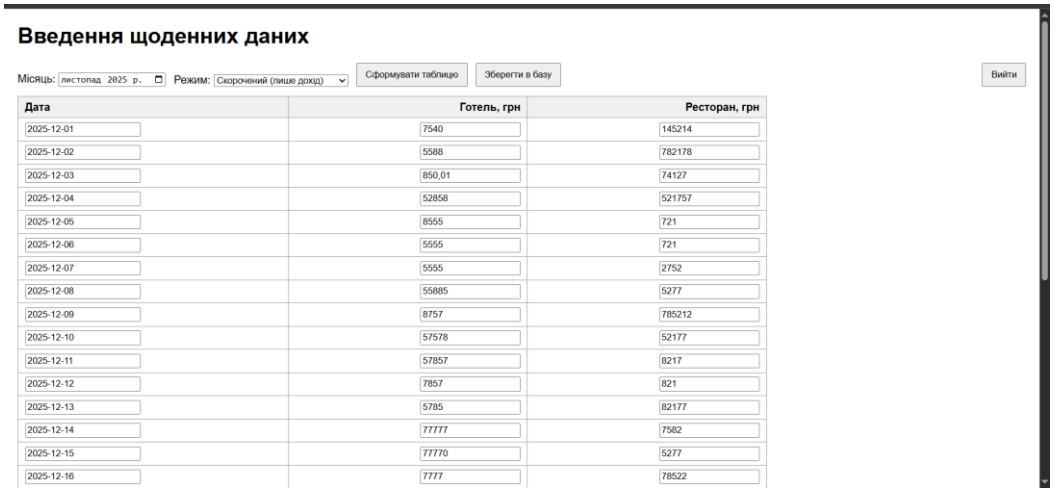


Рис. 3.12 Інтерфейс введення щоденних даних у скороченому режимі

Рис. 3.13 демонструє формування звіту за місяць: агрегована таблиця за період та лінійний графік динаміки. Такий формат подання дозволяє одночасно бачити підсумки та поведінку показників у часі, що особливо корисно при аналізі «піків» або провалів.

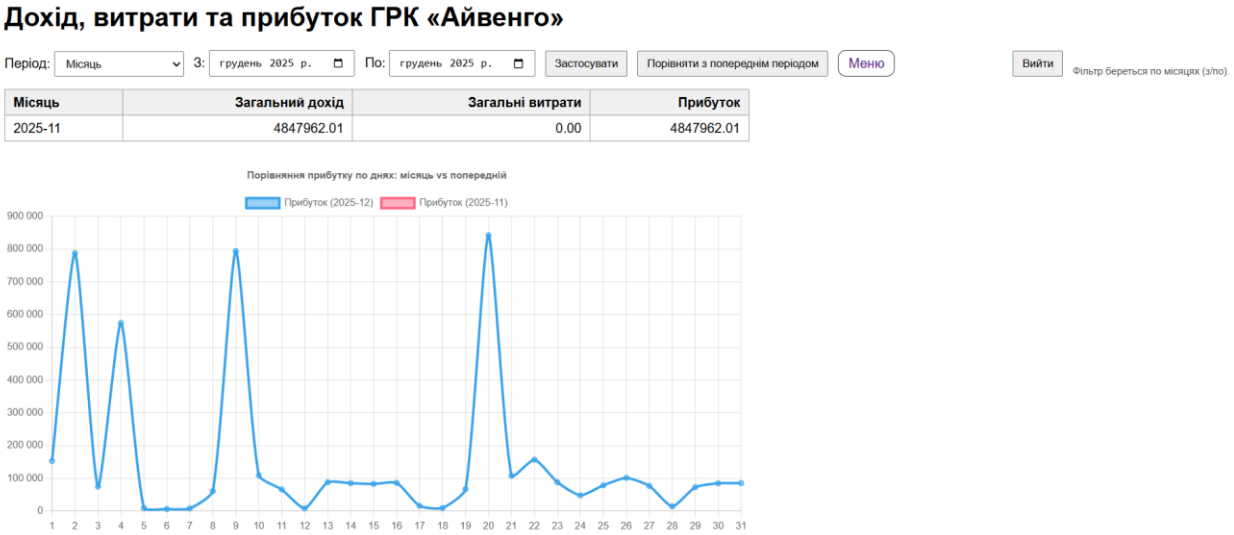


Рис. 3.13 Формування звіту за місяць

Рис. 3.14 відображає звіт за рік із візуалізацією прибутку по місяцях та підказками (tooltip) значень на графіку. Цей сценарій застосовується для оцінки сезонності, виявлення місяців із систематично високими/низькими результатами та порівняння структури року в цілому. Інтерактивні підказки підвищують точність сприйняття: користувач отримує конкретне значення без необхідності додаткових обчислень.

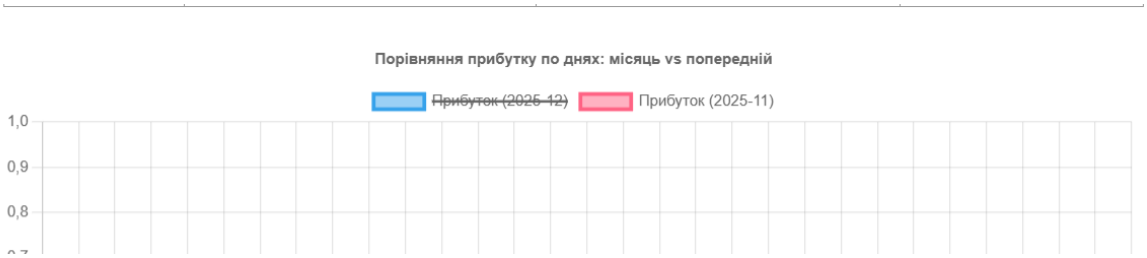
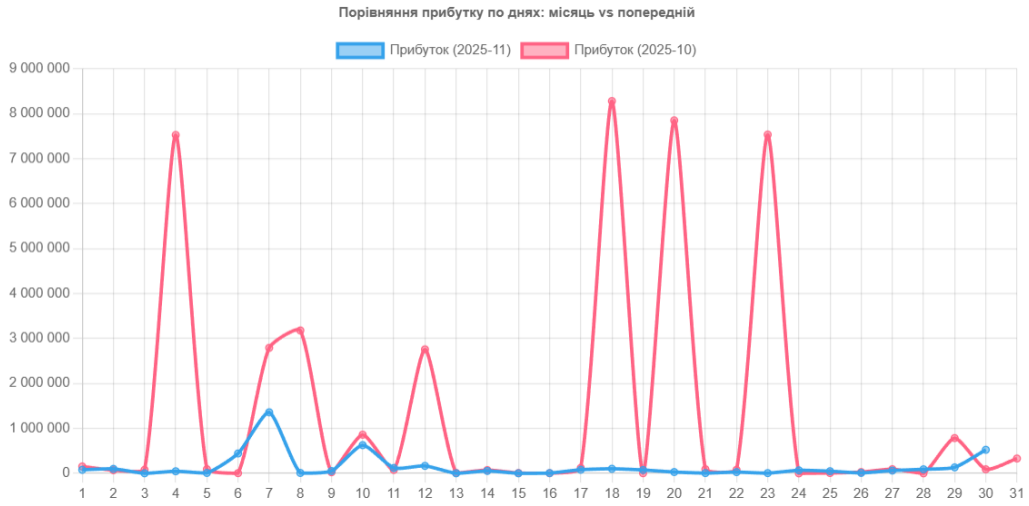


Рис. 3.14 Формування звіту за рік

Наступна ілюстрація демонструє порівняльний режим «поточний місяць vs попередній». На одному графіку відображено дві лінії прибутку по днях, що дає змогу швидко оцінити, як змінюється фінансовий результат у межах календарно співставних відрізків. Під графіком наведено підсумкову таблицю порівняння, яка переводить візуальне спостереження у числові показники.



Порівняння періодів

Період	Сума прибутку	Різниця	%
2025-11	4337011.00	-38681108.00	-89.92 %
2025-10	43018119.00		

Рис. 3.15 Порівняння прибутку по днях

Далі рисунок показує перегляд звіту за кілька місяців із фільтрацією. Такий сценарій корисний, коли потрібно простежити тенденцію без деталізації по днях, але з можливістю зіставлення кількох послідовних періодів у межах одного екрана.

Дохід, витрати та прибуток ГРК «Айвенго»

Період: Місяць З: листопад 2025 р. По: грудень 2025 р. Застосувати Порівняти з попереднім періодом Меню

Місяць	Загальний дохід	Загальні витрати	Прибуток
2025-09	43018119.00	0.00	43018119.00
2025-10	4337011.00	0.00	4337011.00
2025-11	4847962.01	0.00	4847962.01

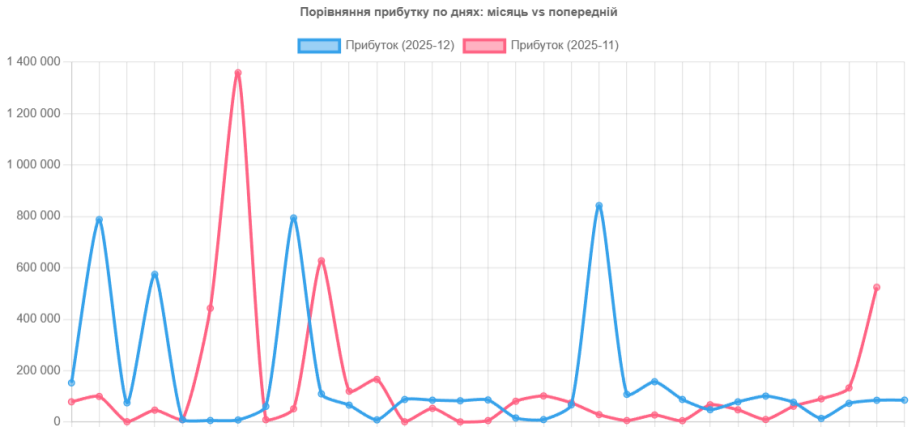


Рис. 3.16 Перегляд звіту за кілька місяців

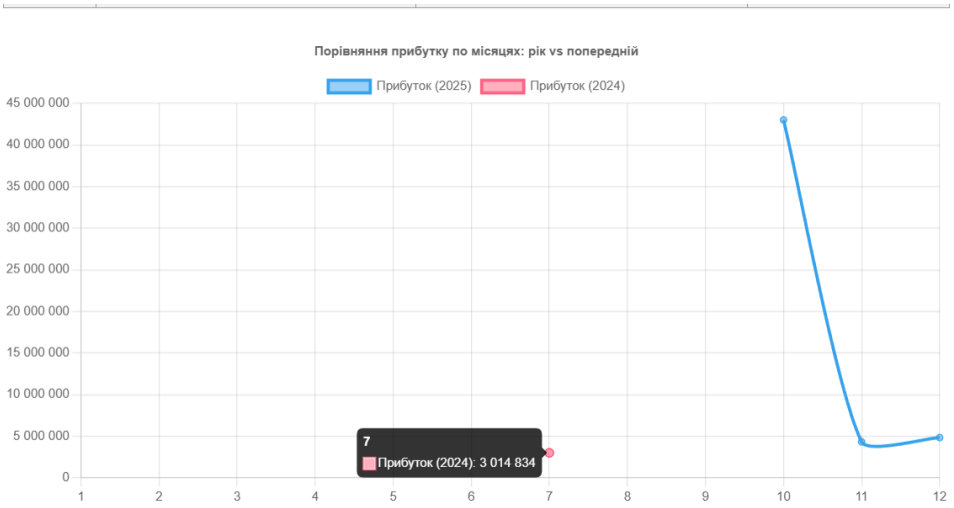
Рис. 3.17 демонструє підсумкову таблицю порівняння періодів, де акцент зроблено на результатах зіставлення: сума прибутку, абсолютна різниця та відносна зміна у відсотках. Цей блок використовується як управлінський підсумок – він швидко відповідає на питання, наскільки поточний період кращий або гірший за базовий.

Порівняння періодів

Період	Сума прибутку	Різниця	%
2025-12	4847962.01	510951.01	11.78 %
2025-11	4337011.00		

Рис. 3.17 Підсумкова таблиця порівняння періодів

І останній рисунок відображає річний графік прибутку з прикладом різких змін динаміки протягом періоду. Такий тип візуалізації підкреслює поведінку показника на довшому горизонті та дозволяє відразу помічати аномалії, після чого керівник може перейти до деталізації та з'ясувати причини.



Порівняння періодів

Період	Сума прибутку	Різниця	%
2025	52203092.01	49188258.01	1631.54 %
2024	3014834.00		

Рис. 3.18 Річний графік прибутку

Окремо варто відзначити, що частина наведених екранів представлена у початкових своїх варіантах і згодом була стилістично змінена. Отже, спочатку перевірялась коректність функціоналу й даних, після чого виконувалась

уніфікація інтерфейсу та покращення візуального сприйняття без зміни алгоритмів агрегації та API.

Висновки по розділу 3

У третьому розділі реалізовано аналітичну інформаційну систему відповідно до розробленого проєкту. Створено та налаштовано базу даних, реалізовано серверну логіку збереження й обробки фінансових даних, а також розроблено web-інтерфейс для введення, аналізу та візуалізації інформації. Проведене тестування підтвердило коректність роботи системи та її здатність формувати аналітичні звіти у різних часових розрізах. Отриманий програмний продукт може бути використаний для підтримки управлінських рішень у діяльності готельно-ресторанного комплексу.

ВИСНОВКИ

У ході виконання магістерської роботи було розглянуто теоретичні, проєктні та практичні аспекти створення аналітичної інформаційної системи для готельно-ресторанного комплексу «Айвенго». Проведене дослідження підтвердило, що в умовах зростання обсягів даних та ускладнення бізнес-процесів у сфері HoReCa традиційні підходи до обліку доходів і витрат є недостатніми для прийняття ефективних управлінських рішень. Саме тому впровадження спеціалізованих аналітичних підсистем є обґрунтованою необхідністю.

У першому розділі проаналізовано особливості діяльності готельно-ресторанних комплексів та визначено роль інформаційних систем у забезпеченні їх конкурентоспроможності. Здійснено огляд сучасних інформаційних систем, що використовуються у готельному та ресторанному бізнесі, а також проведено аналіз існуючого програмного забезпечення, яке застосовується в ГРК «Айвенго». Виявлені обмеження чинних рішень, зокрема відсутність розвиненої аналітики, агрегованої звітності та інструментів порівняння показників у часових розрізах, стали підґрунтям для обґрунтування доцільності розробки окремої аналітичної інформаційної системи.

Другий розділ було присвячено проєктуванню аналітичної системи. Сформульовано основні завдання та вимоги до майбутнього програмного продукту, розроблено структуру даних для аналітичної обробки фінансових показників, а також побудовано модель даних і ER-діаграму бази даних. Запроєктована модель забезпечує цілісність даних, логічну узгодженість між доходами та витратами, а також можливість формування агрегованих показників за місяцями, роками та довільними періодами часу.

У третьому розділі реалізовано програмну частину аналітичної інформаційної системи з використанням сучасних web-технологій.

Розроблено серверну логіку на базі Node.js та Express, створено й налаштовано базу даних MySQL, реалізовано механізми збереження щоденних фінансових даних та формування аналітичних представлень (views). Окрему увагу приділено реалізації інтерфейсу web-додатку, який забезпечує зручне введення даних, перегляд аналітичних звітів і візуалізацію показників у вигляді таблиць та графіків. Проведене тестування підтвердило коректність роботи системи та відповідність поставленим вимогам.

Таким чином, мету магістерської роботи досягнуто – розроблено аналітичну інформаційну систему, яка забезпечує ефективний збір, зберігання та аналіз фінансових показників діяльності готельно-ресторанного комплексу. Запропоноване рішення може бути використане як практичний інструмент для підтримки управлінських рішень, а також має потенціал для подальшого розширення функціональних можливостей, зокрема інтеграції прогнозної аналітики та автоматизованого формування управлінських рекомендацій.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. B52 Software. Програмний комплекс для автоматизації готельно-ресторанного бізнесу. – [Електронний ресурс]. URL: http://b52.biz.ua/ (дата звернення: 04.09.2025).

2. Buhalis D., Leung R. Smart hospitality – interconnectivity and interoperability towards an ecosystem // *International Journal of Hospitality Management*. – 2018. – Vol. 71. – P. 41–50.

3. Chart.js Documentation. Simple yet flexible JavaScript charting library. – [Електронний ресурс]. URL: https://www.chartjs.org/docs/ (дата звернення: 10.09.2025).

4. ChatGPT (GPT-5). OpenAI. – 2025.

5. Express.js Documentation. Fast, unopinionated, minimalist web framework for Node.js. – [Електронний ресурс]. URL: https://expressjs.com/ (дата звернення: 10.09.2025).

6. GoIT Global. Бізнес-аналітика: сутність та практичне застосування. – [Електронний ресурс]. URL: https://goit.global/ua/articles/shcho-take-biznes-analityka-i-chym-vona-korysna-dlia-kompanii/
(дата звернення: 07.09.2025).

7. Hotelogix. Property Management System for Hotels. – [Електронний ресурс]. URL: https://www.hotelogix.com/ (дата звернення: 02.09.2025).

8. Ivanov S., Webster C. *Robotics in Tourism and Hospitality: Concepts, Applications and Implications*. – Springer, 2019.

9. MySQL 8.0 Reference Manual. Oracle Corporation. – [Електронний ресурс]. URL: https://dev.mysql.com/doc/refman/8.0/en/ (дата звернення: 10.09.2025).

10. Node.js Documentation. Node.js JavaScript runtime. – [Електронний ресурс]. URL: https://nodejs.org/en/docs (дата звернення: 10.09.2025).

11. O'Connor P. Room reservations and the internet: implications for hotel distribution // *International Journal of Hospitality Management*. – 2016. – Vol. 48. – P. 10–19.

12. SAP. Role of business analytics in driving change. – [Електронний ресурс]. URL: https://www.sap.com/ukraine/resources/role-of-business-analytics-in-driving-change (дата звернення: 07.09.2025).

13. Socrates VSAU. Аналіз інформаційних систем у готельно-ресторанному бізнесі. – [Електронний ресурс]. URL: https://socrates.vsau.org/b04213/html/cards/getfile.php/17661.pdf (дата звернення: 05.09.2025).

14. Whitesales. The significance of data analytics in business decision-making. – [Електронний ресурс]. URL: https://www.whitesales.ua/uk/blog/unlocking-success-the-significance-of-data-analytics-in-business-decision-making(дата звернення: 07.09.2025).

15. Бізнес-аналітика та її користь для компанії. – [Електронний ресурс]. URL:https://online.novaposhta.education/blog/biznes-analitika-ta-ii-korist-dlya-kompanii (дата звернення: 07.09.2025).

16. Готельно-ресторанний бізнес: організація та управління / Нечаук І. В. – [Електронний ресурс]. URL: https://tourlib.net/books_ukr/nechauk124.htm
(дата звернення: 06.09.2025).

17. ДСТУ 3008:2015. Звіти у сфері науки і техніки. Структура і правила оформлювання. – Київ : ДП «УкрНДНЦ», 2016. – 32 с.

18. ДСТУ 8302:2015. Бібліографічне посилання. Загальні положення та правила складання. – Київ : ДП «УкрНДНЦ», 2016. – 16 с.

19. Інформаційні системи в готельно-ресторанній справі. – [Електронний ресурс]. URL: https://vnu-taskid841251.s3.eu-north-1.amazonaws.com/s3fs-public/File/2023/02/НД%20Інформаційні%20системи%20і%20технології%20в%20готельно-ресторанній%20справі.pdf (дата звернення: 06.09.2025).

20. Інформаційні системи і технології у сфері обслуговування. Освітня програма. – [Електронний ресурс]. URL: https://econom.udpu.edu.ua/wp-content/uploads/files/sert/18032021/3.Бакалавр/ОПП%20Готельно-ресторанна%20справа/Робочі%20програми%202024-2025%20н.р./ОК.33%20Інформаційні%20системи%20і%20технології%20у%20сфері%20обслуговування.pdf (дата звернення: 06.09.2025).

21. Історія розвитку інформаційних систем у сфері гостинності. – [Електронний ресурс]. URL: https://vseosvita.ua/lesson/istoriia-rozvytku-informatsiinykh-system-u-sferi-hostynnosti-352432.html (дата звернення: 05.09.2025).

22. Офіційний сайт ГРК «Айвенго». – [Електронний ресурс]. URL: https://www.aivengo.rv.ua/ (дата звернення: 02.09.2025).

23. Репозиторій КПІ. Сучасні інформаційні системи в управлінні підприємствами гостинності. – [Електронний ресурс]. URL: https://repository.kpi.kharkov.ua/server/api/core/bitstreams/60df456a-560e-46c3-8c22-771057091dde/content (дата звернення: 06.09.2025).

ДОДАТКИ

admin.html:

```
<!DOCTYPE html>
<html lang="uk">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <title>Меню адміністратора</title>
    <link rel="stylesheet" href="/style.css" />
    <style>
      .grid {
        display: grid;
        grid-template-columns: repeat(12, 1fr);
        gap: 16px;
      }
      .hero {
        grid-column: 1 / -1;
        display:flex;
        align-items:flex-end;
        justify-content:space-between;
        gap: 16px;
      }
      .hero h1 { margin:0; }
      .hero .muted { margin-top: 6px; }

      .tile {
        grid-column: span 6;
        padding: 22px;
        border-radius: 18px;
        background: rgba(20, 20, 20, 0.70);
        backdrop-filter: blur(14px);
        -webkit-backdrop-filter: blur(14px);
        box-shadow: 0 18px 36px rgba(0,0,0,.35), inset 0 0 0 1px rgba(255,255,255,.05);
        transition: transform .2s ease, box-shadow .2s ease;
      }
      .tile:hover { transform: translateY(-2px); box-shadow: 0 24px 46px rgba(0,0,0,.45), inset 0 0
0 1px rgba(255,255,255,.06); }

      .tile h2 { margin:0 0 8px; font-size: 18px; }
      .tile p { margin:0 0 14px; opacity:.8; line-height:1.5; }

      .tile .actions { display:flex; gap:10px; flex-wrap:wrap; align-items:center; }
      .chip {
        font-size: 12px;
        padding: 6px 10px;
        border-radius: 999px;
        background: rgba(255,255,255,.08);
        border: 1px solid rgba(255,255,255,.12);
```

```
      opacity: .9;
    }

    @media (max-width: 800px) {
      .tile { grid-column: span 12; }
    }
  </style>
 </head>
 <body>
  <div class="page">
    <div class="card hero">
      <div>
        <h1>Адміністративна панель</h1>
        <div class="muted">Оберіть дію: введення даних або перегляд аналітики</div>
      </div>
      <form method="post" action="/logout">
        <button type="submit" class="secondary">Вийти</button>
      </form>
    </div>

    <div class="grid">
      <div class="tile">
        <div style="display:flex; justify-content:space-between; gap:12px; align-items:flex-
start;">
          <div>
            <h2>Введення щоденних даних</h2>
            <p>Дохід/витрати по днях (скорочений та детальний режим)</p>
          </div>
          <div class="chip">Data Entry</div>
        </div>
        <div class="actions">
          <a class="btnlink" href="/entry.html">Перейти до введення</a>
          <span class="muted"></span>
        </div>
      </div>

      <div class="tile">
        <div style="display:flex; justify-content:space-between; gap:12px; align-items:flex-
start;">
          <div>
            <h2>Аналітичні звіти</h2>
            <p>Перегляд прибутку/витрат/доходу за місяць, рік та 10 років</p>
          </div>
          <div class="chip">Analytics</div>
        </div>
        <div class="actions">
          <a class="btnlink" href="/report.html">Відкрити звіти</a>
          <span class="muted">Графіки</span>
        </div>
      </div>
    </div>
  </div>
```

```
  </body>
</html>
```

entry.html:

```
<!DOCTYPE html>
<html lang="uk">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <title>Введення даних</title>
    <link rel="stylesheet" href="/style.css" />
    <style>
      .topbar {
        display:flex;
        gap: 14px;
        align-items:end;
        flex-wrap:wrap;
        justify-content:space-between;
      }
      .controls {
        display:flex;
        gap: 12px;
        align-items:end;
        flex-wrap:wrap;
      }
      .controls label { display:flex; flex-direction:column; gap:6px; font-size:13px; opacity:.9; }
      .controls .row { display:flex; gap:12px; flex-wrap:wrap; align-items:end; }

      .status {
        margin-top: 12px;
        padding: 10px 12px;
        border-radius: 12px;
        display:none;
        font-size: 13px;
        border: 1px solid rgba(255,255,255,.12);
        background: rgba(255,255,255,.06);
      }
      .status.ok { border-color: rgba(46,204,113,.35); background: rgba(46,204,113,.10);
color:#b7ffcf; display:block; }
      .status.err { border-color: rgba(231,76,60,.35); background: rgba(231,76,60,.10);
color:#ffb3ad; display:block; }
      .status.wait { display:block; opacity:.9; }

      .table-wrap {
        overflow: auto;
        border-radius: 16px;
        border: 1px solid rgba(255,255,255,.08);
      }
      table input[type="number"], table input[type="text"] {
        width: 100%;
        min-width: 120px;
      }
```

```
    table td, table th { white-space: nowrap; }
    .table-wrap { width: 100%; }
    table { width: 100%; table-layout: fixed; }
    th, td { white-space: normal; }
    table input[type="number"], table input[type="text"] {
    min-width: 0 !important;
    width: 100%;
}

  </style>
 </head>

 <body>
  <div class="page">
   <div class="card">
    <div class="topbar">
     <div>
      <h1 style="margin:0;">Введення щоденних даних</h1>
      <div class="muted">Заповнюйте показники та зберігайте їх у базу даних</div>
     </div>

     <div style="display:flex; gap:10px; align-items:center;">
      <a class="btnlink" href="/admin.html">Меню</a>
      <form method="post" action="/logout">
       <button type="submit" class="secondary">Вийти</button>
      </form>
     </div>
    </div>

    <div class="controls" style="margin-top:16px;">
     <label>
      Місяць
      <input type="month" id="monthPicker" />
     </label>

     <label>
      Режим
      <select id="mode">
       <option value="short">Скорочений (лише дохід)</option>
       <option value="full">Детальний (дохід + витрати)</option>
      </select>
     </label>

     <div class="row">
      <button id="generateBtn" type="button">Сформувати таблицю</button>
      <button id="saveBtn" type="button">Зберегти в базу</button>
     </div>
    </div>

    <div class="muted" style="margin-top:10px;">
    </div>
   </div>
```

```
  <div class="card">
    <h2 style="margin-top:0;">Таблиця даних</h2>
    <div class="table-wrap" id="tableContainer"></div>
    <div id="status" class="status"></div>
  </div>
</div>

<script>
  document.addEventListener('DOMContentLoaded', () => {
    const monthPicker = document.getElementById('monthPicker');
    const modeSelect = document.getElementById('mode');
    const generateBtn = document.getElementById('generateBtn');
    const saveBtn = document.getElementById('saveBtn');
    const container = document.getElementById('tableContainer');
    const statusEl = document.getElementById('status');

    function setStatus(type, text) {
      statusEl.className = 'status ' + (type || '');
      statusEl.textContent = text || '';
      if (!text) statusEl.style.display = 'none';
    }

    function cleanNumber(value) {
      if (typeof value === 'string') return value.replace(',', '.');
      return value;
    }

    generateBtn.addEventListener('click', () => {
      const monthVal = monthPicker.value;
      if (!monthVal) return alert('Оберіть місяць');
      const [year, month] = monthVal.split('-').map(Number);
      buildTable(year, month, modeSelect.value);
    });

    monthPicker.addEventListener('change', () => {
      const monthVal = monthPicker.value;
      if (!monthVal) return;
      const [year, month] = monthVal.split('-').map(Number);
      buildTable(year, month, modeSelect.value);
    });

    modeSelect.addEventListener('change', () => {
      const monthVal = monthPicker.value;
      if (!monthVal) {
        toggleExpenseColumns(modeSelect.value);
        return;
      }
      const [year, month] = monthVal.split('-').map(Number);
      buildTable(year, month, modeSelect.value);
    });
```

```
      function buildTable(year, month, mode) {
        const daysInMonth = new Date(year, month, 0).getDate();

        const table = document.createElement('table');

        const thead = document.createElement('thead');
        thead.innerHTML = `
          <tr>
            <th>Дата</th>
            <th>Готель, грн</th>
            <th>Ресторан, грн</th>
            <th class="exp-col">Електроенергія</th>
            <th class="exp-col">Продукти</th>
            <th class="exp-col">Зарплата</th>
            <th class="exp-col">Інші витрати</th>
            <th class="exp-col">Обслуговування</th>
          </tr>
        `;
        table.appendChild(thead);

        const tbody = document.createElement('tbody');

        for (let day = 1; day <= daysInMonth; day++) {
          const dateStr = `${year}-${String(month).padStart(2, '0')}-${String(day).padStart(2,
'0')}`;

          const tr = document.createElement('tr');
          tr.innerHTML = `
            <td><input type="text" value="${dateStr}" readonly /></td>
            <td><input type="number" step="0.01" value="0" class="hotel" /></td>
            <td><input type="number" step="0.01" value="0" class="restaurant" /></td>
            <td class="exp-col"><input type="number" step="0.01" value="0" class="electricity"
/></td>
            <td class="exp-col"><input type="number" step="0.01" value="0" class="food" /></td>
            <td class="exp-col"><input type="number" step="0.01" value="0" class="salary"
/></td>
            <td class="exp-col"><input type="number" step="0.01" value="0" class="other"
/></td>
            <td class="exp-col"><input type="number" step="0.01" value="0" class="maint"
/></td>
          `;
          tbody.appendChild(tr);
        }

        table.appendChild(tbody);
        container.innerHTML = '';
        container.appendChild(table);

        toggleExpenseColumns(mode);
      }

      function toggleExpenseColumns(mode) {
```

```
    const hide = mode === 'short';
    document.querySelectorAll('.exp-col')
      .forEach((td) => (td.style.display = hide ? 'none' : 'table-cell'));
  }

  saveBtn.addEventListener('click', async () => {
    const table = container.querySelector('table');
    if (!table) return alert('Спочатку сформуйте таблицю');

    const trs = table.querySelectorAll('tbody tr');
    const shortMode = modeSelect.value === 'short';

    const rows = [];
    trs.forEach((tr) => {
      const date = tr.querySelector('td:nth-child(1) input').value;

      const hotel = Number(cleanNumber(tr.querySelector('td:nth-child(2) input').value) || '0');
      const restaurant = Number(cleanNumber(tr.querySelector('td:nth-child(3) input').value) ||
'0');

      let electricity = 0, food = 0, salary = 0, other = 0, maint = 0;

      if (!shortMode) {
        electricity = Number(cleanNumber(tr.querySelector('td:nth-child(4) input').value) || '0');
        food = Number(cleanNumber(tr.querySelector('td:nth-child(5) input').value) || '0');
        salary = Number(cleanNumber(tr.querySelector('td:nth-child(6) input').value) || '0');
        other = Number(cleanNumber(tr.querySelector('td:nth-child(7) input').value) || '0');
        maint = Number(cleanNumber(tr.querySelector('td:nth-child(8) input').value) || '0');
      }

      rows.push({
        date,
        hotel_revenue: hotel,
        restaurant_revenue: restaurant,
        electricity_cost: electricity,
        food_cost: food,
        salary_cost: salary,
        other_costs: other,
        maintenance_cost: maint,
      });
    });

    setStatus('wait', 'Збереження...');

    try {
      const payload = { rows, mode: modeSelect.value };
      const resp = await fetch('/api/daily-data', {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify(payload),
      });
```

```
        if (!resp.ok) {
          const txt = await resp.text();
          setStatus('err', `Помилка збереження: ${resp.status} - ${txt}`);
          return;
        }

        const data = await resp.json();
        if (data.ok) setStatus('ok', 'Дані успішно збережено ');
        else setStatus('err', 'Помилка збереження: ' + (data.error || 'невідомо'));
      } catch (err) {
        console.error(err);
        setStatus('err', 'Помилка збереження (мережа або сервер)');
      }
    });

    const today = new Date();
    const yyyy = today.getFullYear();
    const monthNum = today.getMonth() + 1;
    const mm = String(monthNum).padStart(2, '0');
    monthPicker.value = `${yyyy}-${mm}`;
    buildTable(yyyy, monthNum, modeSelect.value);
  });
  </script>
 </body>
</html>
```

login.html:

```
<!DOCTYPE html>
<html lang="uk">
 <head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1" />
  <title>Вхід адміністратора</title>
  <link rel="stylesheet" href="/style.css" />
  <style>
   .center {
    min-height: 100vh;
    display: grid;
    place-items: center;
    padding: 24px;
   }
   .auth-card {
    width: min(420px, 100%);
   }
   .brand {
    display:flex;
    align-items:center;
    justify-content:space-between;
    gap: 14px;
    margin-bottom: 14px;
   }
   .badge {
```

```
      font-size: 12px;
      padding: 6px 10px;
      border-radius: 999px;
      background: rgba(255,255,255,.08);
      border: 1px solid rgba(255,255,255,.12);
      opacity: .9;
    }
    .field { margin-top: 12px; }
    .field label { display:block; font-size: 13px; opacity: .85; margin-bottom: 6px; }
    .field input { width: 100%; }
    .actions {
      margin-top: 16px;
      display:flex;
      gap: 10px;
      align-items:center;
    }
    .hint {
      margin-top: 12px;
      opacity: .75;
      font-size: 12px;
      line-height: 1.4;
    }
    .error {
      display:none;
      margin-top: 12px;
      padding: 10px 12px;
      border-radius: 12px;
      background: rgba(231,76,60,.12);
      border: 1px solid rgba(231,76,60,.25);
      color: #ffb3ad;
      font-size: 13px;
    }
  </style>
 </head>
 <body>
  <div class="center">
   <div class="card auth-card">
    <div class="brand">
     <div>
      <h1 style="margin:0; font-size:22px;">AYVENGO Analytics</h1>
      <div class="muted">Панель адміністратора</div>
     </div>
     <div class="badge">Secure Login</div>
    </div>

    <form method="post" action="/login" id="loginForm">
     <div class="field">
      <label for="username">Логін</label>
      <input id="username" type="text" name="username" autocomplete="username" required
/>
     </div>
```

```
        <div class="field">
          <label for="password">Пароль</label>
          <input id="password" type="password" name="password" autocomplete="current-
password" required />
        </div>

        <div class="actions">
          <button type="submit">Увійти</button>
          <span class="muted">Доступ для адміністратора</span>
        </div>

        <div class="error" id="errBox"></div>

        <div class="hint">
        </div>
      </form>
    </div>
  </div>

  <script>

    const form = document.getElementById('loginForm');
    const errBox = document.getElementById('errBox');

    form.addEventListener('submit', async (e) => {

      e.preventDefault();
      errBox.style.display = 'none';
      errBox.textContent = '';

      const fd = new FormData(form);
      const resp = await fetch('/login', { method: 'POST', body: new URLSearchParams(fd) });

      if (resp.redirected) {
        window.location.href = resp.url;
        return;
      }

      if (!resp.ok) {
        const txt = await resp.text();
        errBox.textContent = txt || 'Помилка входу';
        errBox.style.display = 'block';
        return;
      }

      // fallback
      window.location.href = '/admin.html';
    });
  </script>
 </body>
</html>
```

report.html:

```
<!DOCTYPE html>
<html lang="uk">
 <head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1" />
  <title>Звіт Айвенго – графіки</title>
  <link rel="stylesheet" href="/style.css" />
  <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
  <style>
   #filters { display:flex; gap:12px; align-items:end; flex-wrap:wrap; }
   #filters label{ display:flex; flex-direction:column; gap:6px; font-size:13px; opacity:.95; }
   .right { margin-left:auto; display:flex; gap:10px; align-items:center; flex-wrap:wrap; }
   .hint { margin-top: 10px; }
   canvas { width:100% !important; height: 420px !important; }
   @media (max-width: 800px){
    canvas { height: 340px !important; }
   }
  </style>
 </head>

 <body>
  <div class="page">
   <div class="card">
    <h1>Дохід, витрати та прибуток ГРК «Айвенго»</h1>

    <div id="filters">
     <label>
      Період:
      <select id="period">
       <option value="month">Місяць</option>
       <option value="year">Рік</option>
       <option value="10y">10 років (по роках)</option>
      </select>
     </label>

     <label id="fromWrap">
      <input type="month" id="fromVal" />
     </label>

     <label id="toWrap">
      По:
      <input type="month" id="toVal" />
     </label>

     <button id="applyBtn" type="button">Застосувати</button>
     <button id="compareBtn" type="button">Порівняти з попереднім періодом</button>

     <div class="right">
      <a class="btnlink" href="/admin.html">Меню</a>
      <form method="post" action="/logout">
       <button type="submit" class="secondary">Вийти</button>
      </form>
```

```
      </div>
    </div>

    <div class="muted hint" id="hint"></div>
  </div>

  <div class="card">
    <div class="table-wrap">
      <table id="profit-table">
        <thead>
          <tr>
            <th id="colPeriod">Місяць</th>
            <th>Загальний дохід</th>
            <th>Загальні витрати</th>
            <th>Прибуток</th>
          </tr>
        </thead>
        <tbody></tbody>
      </table>
    </div>

    <div style="margin-top:18px;">
      <canvas id="profitChart"></canvas>
    </div>
  </div>

  <div class="card" id="compareSummary" style="display:none;">
    <h2>Порівняння періодів</h2>
    <div class="table-wrap">
      <table>
        <thead>
          <tr>
            <th>Період</th>
            <th>Сума прибутку</th>
            <th>Різниця</th>
            <th>%</th>
          </tr>
        </thead>
        <tbody id="compareSummaryBody"></tbody>
      </table>
    </div>
  </div>
</div>

<script>
  const tbody = document.querySelector('#profit-table tbody');
  const periodSel = document.getElementById('period');
  const fromWrap = document.getElementById('fromWrap');
  const toWrap = document.getElementById('toWrap');
  const fromVal = document.getElementById('fromVal');
  const toVal = document.getElementById('toVal');
  const applyBtn = document.getElementById('applyBtn');
```

```
const compareBtn = document.getElementById('compareBtn');
const colPeriod = document.getElementById('colPeriod');
const hint = document.getElementById('hint');

let chart = null;

function setUiByPeriod() {
  const p = periodSel.value;
  compareBtn.disabled = (p === '10y');

  if (p === '10y') {
    fromWrap.style.display = 'none';
    toWrap.style.display = 'none';
    colPeriod.textContent = 'Рік';
    hint.textContent = 'Показує останні 10 років (по роках).';
    return;
  }

  fromWrap.style.display = '';
  toWrap.style.display = '';
  colPeriod.textContent = (p === 'year') ? 'Рік' : 'Місяць';
  hint.textContent = (p === 'year')
    ? 'Фільтр береться по роках (з/по).'
    : 'Фільтр береться по місяцях (з/по).';
}

function fmtMoney(n) {
  const x = Number(n) || 0;
  return x.toFixed(2);
}

function sumArray(arr) {
  return arr.reduce((s, v) => s + (Number(v) || 0), 0);
}

async function loadData() {
  // hide compare block on normal load
  const cs = document.getElementById('compareSummary');
  if (cs) cs.style.display = 'none';

  const p = periodSel.value;
  const params = new URLSearchParams();
  params.set('period', p);

  if (p !== '10y') {
    if (fromVal.value) params.set('from', fromVal.value);
    if (toVal.value) params.set('to', toVal.value);
  }

  const resp = await fetch('/api/profit?' + params.toString());
  const data = await resp.json();
  render(data, p);
```

```
}

function render(data, p) {
  tbody.innerHTML = '';

  const labels = [];
  const revenues = [];
  const expenses = [];
  const profits = [];

  data.forEach((row) => {
    const key = (p === 'month') ? row.month : row.year_start;
    const label = String(key).slice(0, 10);

    const revenue = Number(row.total_revenue) || 0;
    const expense = Number(row.total_expenses) || 0;
    const profit = Number(row.profit) || 0;

    labels.push(label);
    revenues.push(revenue);
    expenses.push(expense);
    profits.push(profit);

    const tr = document.createElement('tr');
    tr.innerHTML = `
      <td>${p === 'month' ? label.slice(0, 7) : label.slice(0, 4)}</td>
      <td>${fmtMoney(revenue)}</td>
      <td>${fmtMoney(expense)}</td>
      <td class="${profit < 0 ? 'negative' : 'positive'}">${fmtMoney(profit)}</td>
    `;
    tbody.appendChild(tr);
  });

  drawChart(
    labels.map(x => (p === 'month' ? x.slice(0, 7) : x.slice(0, 4))),
    revenues, expenses, profits
  );
}

function drawChart(labels, revenue, expenses, profit) {
  const ctx = document.getElementById('profitChart').getContext('2d');
  if (chart) chart.destroy();

  chart = new Chart(ctx, {
    type: 'line',
    data: {
      labels,
      datasets: [
        {
          label: 'Доходи',
          data: revenue,
          tension: 0.3,
```

```
        borderWidth: 2,
        borderColor: '#2ecc71',
        backgroundColor: 'rgba(46,204,113,0.12)',
        fill: true
       },
       {
        label: 'Витрати',
        data: expenses,
        tension: 0.3,
        borderWidth: 2,
        borderColor: '#e74c3c',
        backgroundColor: 'rgba(231,76,60,0.10)',
        fill: true
       },
       {
        label: 'Прибуток',
        data: profit,
        tension: 0.3,
        borderWidth: 2,
        borderColor: '#f1c40f',
        backgroundColor: 'rgba(241,196,15,0.10)',
        fill: true
       }
     ]
    },
    options: {
     responsive: true,
     plugins: {
      legend: { labels: { color: '#eaeaea' } }
     },
     scales: {
      x: { ticks: { color: '#cfcfcf' }, grid: { color: 'rgba(255,255,255,.06)' } },
      y: { ticks: { color: '#cfcfcf' }, grid: { color: 'rgba(255,255,255,.06)' } }
     }
    }
  });
}

async function loadCompare() {
  const p = periodSel.value;
  if (p === '10y') {
    alert('Для «10 років» порівняння не використовується.');
    return;
  }

  let url = '';
  if (p === 'month') {
    const month = (toVal.value || fromVal.value);
    if (!month) return alert('Оберіть місяць у фільтрі.');
    url = `/api/compare?period=month&month=${encodeURIComponent(month)}`;
  } else {
    const year = (toVal.value || fromVal.value || '').slice(0, 4);
```

```
        if (!/^\d{4}$/.test(year)) return alert('Оберіть рік у фільтрі.');
        url = `/api/compare?period=year&year=${encodeURIComponent(year)}`;
      }

      const resp = await fetch(url);
      if (!resp.ok) {
        const txt = await resp.text();
        alert('Помилка порівняння: ' + txt);
        return;
      }

      const data = await resp.json();

      drawCompareChart(data.labels, data.currentProfit, data.previousProfit, data.currentLabel,
data.previousLabel, p);
      renderCompareSummary(data.currentProfit, data.previousProfit, data.currentLabel,
data.previousLabel);
    }

    function drawCompareChart(labels, curProfit, prevProfit, curLabel, prevLabel, p) {
      const ctx = document.getElementById('profitChart').getContext('2d');
      if (chart) chart.destroy();

      const xLabels = labels.map(v => String(v));
      chart = new Chart(ctx, {
        type: 'line',
        data: {
          labels: xLabels,
          datasets: [
            { label: `Прибуток (${curLabel})`, data: curProfit, tension: 0.3, borderWidth: 2,
borderColor: '#2ecc71' },
            { label: `Прибуток (${prevLabel})`, data: prevProfit, tension: 0.3, borderWidth: 2,
borderColor: '#e74c3c' }
          ]
        },
        options: {
          responsive: true,
          plugins: {
            legend: { labels: { color: '#eaeaea' } },
            title: {
              display: true,
              text: p === 'month'
                ? 'Порівняння прибутку по днях: місяць vs попередній'
                : 'Порівняння прибутку по місяцях: рік vs попередній',
              color: '#eaeaea'
            }
          },
          scales: {
            x: { ticks: { color: '#cfcfcf' }, grid: { color: 'rgba(255,255,255,.06)' } },
            y: { ticks: { color: '#cfcfcf' }, grid: { color: 'rgba(255,255,255,.06)' } }
          }
        }
```

```
      });
    }

    function renderCompareSummary(curArr, prevArr, curLabel, prevLabel) {
      const wrap = document.getElementById('compareSummary');
      const body = document.getElementById('compareSummaryBody');

      const curSum = sumArray(curArr);
      const prevSum = sumArray(prevArr);
      const diff = curSum - prevSum;
      const percent = prevSum !== 0 ? (diff / prevSum) * 100 : 0;

      body.innerHTML = `
        <tr>
          <td>${curLabel}</td>
          <td>${fmtMoney(curSum)}</td>
          <td rowspan="2" class="${diff < 0 ? 'negative' : 'positive'}">${fmtMoney(diff)}</td>
          <td rowspan="2" class="${diff < 0 ? 'negative' : 'positive'}">${percent.toFixed(2)}
%</td>
        </tr>
        <tr>
          <td>${prevLabel}</td>
          <td>${fmtMoney(prevSum)}</td>
        </tr>
      `;

      wrap.style.display = 'block';
    }

    const now = new Date();
    const yyyy = now.getFullYear();
    const mm = String(now.getMonth() + 1).padStart(2, '0');
    fromVal.value = `${yyyy}-${mm}`;
    toVal.value = `${yyyy}-${mm}`;

    periodSel.addEventListener('change', () => {
      setUiByPeriod();
      loadData();
    });

    applyBtn.addEventListener('click', loadData);
    compareBtn.addEventListener('click', loadCompare);

    setUiByPeriod();
    loadData();
  </script>
 </body>
</html>
```

style.css:

```
* { box-sizing: border-box; }
```

```
:root{
  --bg: #0b0b0b;
  --card: rgba(10,10,10,.78);
  --stroke: rgba(255,255,255,.08);
  --stroke2: rgba(255,255,255,.12);
  --text: #f2f2f2;
  --muted: rgba(255,255,255,.75);
  --green: #2ecc71;
  --green2:#27ae60;
  --red: #e74c3c;
  --red2:#c0392b;
}

body{
  margin:0;
  min-height:100vh;
  font-family: Inter, "Segoe UI", system-ui, Arial, sans-serif;
  color: var(--text);

  background:
    radial-gradient(900px 450px at 20% 10%, rgba(46,204,113,.12), transparent 60%),
    radial-gradient(900px 450px at 80% 20%, rgba(231,76,60,.12), transparent 60%),
    linear-gradient(rgba(0,0,0,.68), rgba(0,0,0,.68)),
    url("/bg-analytics.png") center/cover no-repeat fixed;
}

/* контейнер сторінок */
.page{
  max-width: 1200px;
  margin: 0 auto;
  padding: 32px 24px 60px;
}

h1,h2,h3,th,td,label,.muted{
  text-shadow: 0 1px 10px rgba(0,0,0,.45);
}

h1{ font-size: 32px; margin: 0 0 10px; font-weight: 700; letter-spacing:.3px; }
h2{ font-size: 20px; margin: 0 0 10px; font-weight: 650; letter-spacing:.2px; }
h3{ font-size: 16px; margin: 0 0 10px; font-weight: 650; letter-spacing:.2px; }

.muted{ opacity:.85; font-size: 13px; color: var(--muted); }

/* card glass */
.card{
  background: var(--card);
  backdrop-filter: blur(14px);
  -webkit-backdrop-filter: blur(14px);
  border-radius: 18px;
  padding: 20px 22px;
  margin-bottom: 24px;
  box-shadow:
```

```
    0 20px 50px rgba(0,0,0,.55),
    inset 0 0 0 1px rgba(255,255,255,.06);
}

/* buttons */
button, .btnlink{
  border:none;
  border-radius:12px;
  padding: 10px 18px;
  font-size: 14px;
  font-weight: 600;
  cursor: pointer;
  color: #fff;
  background: linear-gradient(135deg, var(--green), var(--green2));
  box-shadow: 0 10px 22px rgba(46,204,113,.30);
  transition: transform .2s ease, box-shadow .2s ease, filter .2s ease;
  text-decoration:none;
  display:inline-block;
}

button:hover, .btnlink:hover{
  transform: translateY(-1px);
  box-shadow: 0 16px 32px rgba(46,204,113,.40);
  filter: brightness(1.02);
}

button:disabled{
  opacity: .5;
  cursor: not-allowed;
  transform: none;
  box-shadow: none;
}

button.secondary{
  background: linear-gradient(135deg, var(--red), var(--red2));
  box-shadow: 0 10px 22px rgba(231,76,60,.28);
}
button.secondary:hover{
  box-shadow: 0 16px 32px rgba(231,76,60,.38);
}

/* inputs */
input, select{
  background: rgba(255,255,255,.08);
  border: 1px solid rgba(255,255,255,.15);
  border-radius: 10px;
  padding: 8px 10px;
  color: #fff;
}
input:focus, select:focus{
  outline:none;
  border-color: var(--green);
```

```
}

/* table */
.table-wrap{
  width:100%;
  overflow:auto;
  border-radius: 16px;
  border: 1px solid var(--stroke);
}

table{
  width:100%;
  border-collapse: collapse;
  font-size: 14px;
  color: var(--text);
}

thead{
  background: rgba(255,255,255,.10);
}

th, td{
  padding: 10px 14px;
  border-bottom: 1px solid rgba(255,255,255,.08);
}

th{
  text-align:left;
  font-weight: 650;
  color: #fff;
  opacity: .95;
}

td{ text-align:right; color: #f1f1f1; }
td:first-child, th:first-child{ text-align:left; }

tr:hover{ background: rgba(255,255,255,.06); }

.negative{ color:#ff6b6b; font-weight: 750; }
.positive{ color: var(--green); font-weight: 750; }
```

.env:

```
DB_HOST=localhost
DB_PORT=3306
DB_USER=root
DB_PASSWORD=Mysql2024!
DB_NAME=ayvengo_analytics
SESSION_SECRET=whatever_long_string
```

db.js:

```
const mysql = require('mysql2/promise');
```

```
require('dotenv').config();

const pool = mysql.createPool({
host: process.env.DB_HOST,
port: Number(process.env.DB_PORT || 3306),
user: process.env.DB_USER,
password: process.env.DB_PASSWORD,
database: process.env.DB_NAME,
waitForConnections: true,
connectionLimit: 10,
queueLimit: 0
});

module.exports = pool;
```

hash.js:

```
const bcrypt = require('bcrypt');

bcrypt.compare(inputPassword, user.password_hash)
  .then(match => {
    if (match) console.log("Login success");
    else console.log("Wrong password");
  });
```

index.js:

```
const express = require('express');
const path = require('path');
const cookieParser = require('cookie-parser');
const bcrypt = require('bcryptjs');
const dotenv = require('dotenv');
const pool = require('./db');

dotenv.config();

const app = express();
const PORT = process.env.PORT || 3000;
const COOKIE_SECRET = process.env.SESSION_SECRET || 'super-secret';

app.use(express.urlencoded({ extended: true }));
app.use(express.json());
app.use(cookieParser(COOKIE_SECRET));

app.use(express.static(path.join(__dirname, 'public')));

function requireAuth(req, res, next) {
  if (req.signedCookies && req.signedCookies.isAdmin === '1') return next();
  return res.redirect('/login');
}

app.get('/login', (req, res) => {
  res.sendFile(path.join(__dirname, 'public', 'login.html'));
});
```

```
app.get('/', (req, res) => {
  if (req.signedCookies && req.signedCookies.isAdmin === '1') return
res.redirect('/admin.html');
  return res.redirect('/login');
});

app.get('/admin.html', requireAuth, (req, res) => {
  res.sendFile(path.join(__dirname, 'public', 'admin.html'));
});

app.get('/entry.html', requireAuth, (req, res) => {
  res.sendFile(path.join(__dirname, 'public', 'entry.html'));
});

app.get('/report.html', requireAuth, (req, res) => {
  res.sendFile(path.join(__dirname, 'public', 'report.html'));
});

app.post('/login', async (req, res) => {
  const { username, password } = req.body;

  try {
    const [rows] = await pool.query(
      'SELECT id, username, password_hash FROM users WHERE username = ? LIMIT 1',
      [username]
    );

    if (rows.length === 0) return res.status(401).send('Невірний логін або пароль');

    const user = rows[0];
    const ok = await bcrypt.compare(password, user.password_hash);
    if (!ok) return res.status(401).send('Невірний логін або пароль');

    res.cookie('isAdmin', '1', {
      httpOnly: true,
      signed: true,
      sameSite: 'lax',
      maxAge: 8 * 60 * 60 * 1000,
    });

    return res.redirect('/admin.html');
  } catch (err) {
    console.error('Помилка /login:', err);
    return res.status(500).send('Внутрішня помилка сервера');
  }
});

app.post('/logout', (req, res) => {
  res.clearCookie('isAdmin');
  res.redirect('/login');
});
```

```
app.get('/api/profit', requireAuth, async (req, res) => {
  const period = String(req.query.period || 'month').toLowerCase(); // month | year | 10y
  const from = req.query.from; // YYYY-MM або YYYY-MM-01
  const to = req.query.to;

  try {
    if (period === 'month') {
      let sql = 'SELECT month, total_revenue, total_expenses, profit FROM monthly_profit';
      const params = [];
      const where = [];

      if (from) {
        where.push('month >= ?');
        params.push(from.slice(0, 7) + '-01');
      }
      if (to) {
        where.push('month <= ?');
        params.push(to.slice(0, 7) + '-01');
      }

      if (where.length) sql += ' WHERE ' + where.join(' AND ');
      sql += ' ORDER BY month';

      const [rows] = await pool.query(sql, params);
      return res.json(rows);
    }

    if (period === 'year') {
      let sql = 'SELECT year_start, total_revenue, total_expenses, profit FROM yearly_profit';
      const params = [];
      const where = [];

      if (from) {
        const y = Number(String(from).slice(0, 4));
        if (!Number.isNaN(y)) {
          where.push('YEAR(year_start) >= ?');
          params.push(y);
        }
      }
      if (to) {
        const y = Number(String(to).slice(0, 4));
        if (!Number.isNaN(y)) {
          where.push('YEAR(year_start) <= ?');
          params.push(y);
        }
      }

      if (where.length) sql += ' WHERE ' + where.join(' AND ');
      sql += ' ORDER BY year_start';

      const [rows] = await pool.query(sql, params);
```

```
    return res.json(rows);
   }

   if (period === '10y') {
    const [rows] = await pool.query(
     'SELECT year_start, total_revenue, total_expenses, profit FROM ten_year_profit ORDER
BY year_start'
    );
    return res.json(rows);
   }

   return res.status(400).json({ error: 'Invalid period. Use month|year|10y' });
  } catch (err) {
   console.error('/api/profit error:', err);
   return res.status(500).json({ error: 'Помилка сервера' });
  }
});

app.get('/api/monthly-profit', requireAuth, async (req, res) => {
  const { from, to } = req.query;
  const params = new URLSearchParams();
  params.set('period', 'month');
  if (from) params.set('from', from);
  if (to) params.set('to', to);

  req.query.period = 'month';
  req.query.from = from;
  req.query.to = to;
  return app._router.handle(req, res, () => {}, 'get', '/api/profit');
});

app.post('/api/daily-data', requireAuth, async (req, res) => {
  const { rows, mode } = req.body;

  if (!Array.isArray(rows) || rows.length === 0 || !mode) {
   return res.status(400).json({ ok: false, error: 'Невірний формат даних' });
  }

  let conn;
  try {
   conn = await pool.getConnection();
   await conn.beginTransaction();

   for (const r of rows) {
    const date = r.date;

    await conn.query(
     `INSERT INTO daily_revenue (date, hotel_revenue, restaurant_revenue)
      VALUES (?, ?, ?)
      ON DUPLICATE KEY UPDATE
       hotel_revenue = VALUES(hotel_revenue),
       restaurant_revenue = VALUES(restaurant_revenue)`,
```

```
      [date, Number(r.hotel_revenue || 0), Number(r.restaurant_revenue || 0)]
    );

    if (mode === 'full') {
     await conn.query(
      `INSERT INTO daily_expenses
       (date, electricity_cost, food_cost, salary_cost, other_costs, maintenance_cost)
       VALUES (?, ?, ?, ?, ?, ?)
       ON DUPLICATE KEY UPDATE
        electricity_cost = VALUES(electricity_cost),
        food_cost = VALUES(food_cost),
        salary_cost = VALUES(salary_cost),
        other_costs = VALUES(other_costs),
        maintenance_cost = VALUES(maintenance_cost)`,
      [
       date,
       Number(r.electricity_cost || 0),
       Number(r.food_cost || 0),
       Number(r.salary_cost || 0),
       Number(r.other_costs || 0),
       Number(r.maintenance_cost || 0),
      ]
     );
    } else {
     await conn.query(
      `INSERT INTO daily_expenses
       (date, electricity_cost, food_cost, salary_cost, other_costs, maintenance_cost)
       VALUES (?, 0, 0, 0, 0, 0)
       ON DUPLICATE KEY UPDATE
        electricity_cost = 0, food_cost = 0, salary_cost = 0, other_costs = 0, maintenance_cost =
0`,
      [date]
     );
    }
   }

   await conn.commit();
   res.json({ ok: true });
  } catch (err) {
   if (conn) await conn.rollback();
   console.error('/api/daily-data error:', err);
   res.status(500).json({ ok: false, error: 'Помилка збереження даних' });
  } finally {
   if (conn) conn.release();
  }
});

function lastDayOfMonth(year, month) {
 return new Date(year, month, 0).getDate();
}

function addMonths(year, month, delta) {
```

```
  const d = new Date(year, month - 1, 1);
  d.setMonth(d.getMonth() + delta);
  return { year: d.getFullYear(), month: d.getMonth() + 1 };
}

app.get('/api/compare', requireAuth, async (req, res) => {
  const period = String(req.query.period || '').toLowerCase();

  try {
    if (period === 'month') {
      const m = String(req.query.month || ''); // YYYY-MM
      if (!/^\d{4}-\d{2}$/.test(m)) {
        return res.status(400).json({ error: 'month must be YYYY-MM' });
      }

      const year = Number(m.slice(0, 4));
      const month = Number(m.slice(5, 7));

      const prev = addMonths(year, month, -1);

      const curDays = lastDayOfMonth(year, month);
      const prevDays = lastDayOfMonth(prev.year, prev.month);
      const maxDays = Math.max(curDays, prevDays);

      const curFrom = `${year}-${String(month).padStart(2, '0')}-01`;
      const curTo = `${year}-${String(month).padStart(2, '0')}-${String(curDays).padStart(2,
'0')}`;

      const prevFrom = `${prev.year}-${String(prev.month).padStart(2, '0')}-01`;
      const prevTo = `${prev.year}-${String(prev.month).padStart(2, '0')}-
${String(prevDays).padStart(2, '0')}`;

      const sqlDay = `
        SELECT
          DAY(dr.date) AS d,
          ((dr.hotel_revenue + dr.restaurant_revenue) -
           (COALESCE(de.electricity_cost,0)+COALESCE(de.food_cost,0)+COALESCE(de.salary
_cost,0)+COALESCE(de.other_costs,0)+COALESCE(de.maintenance_cost,0))) AS profit
        FROM daily_revenue dr
        LEFT JOIN daily_expenses de ON de.date = dr.date
        WHERE dr.date BETWEEN ? AND ?
        ORDER BY dr.date
      `;

      const [curRows] = await pool.query(sqlDay, [curFrom, curTo]);
      const [prevRows] = await pool.query(sqlDay, [prevFrom, prevTo]);

      const labels = Array.from({ length: maxDays }, (_, i) => i + 1);
      const currentProfit = Array(maxDays).fill(null);
      const previousProfit = Array(maxDays).fill(null);

      curRows.forEach(r => { currentProfit[(r.d || 0) - 1] = Number(r.profit) || 0; });
```

```
    prevRows.forEach(r => { previousProfit[(r.d || 0) - 1] = Number(r.profit) || 0; });

    return res.json({
      labels,
      currentLabel: m,
      previousLabel: `${prev.year}-${String(prev.month).padStart(2, '0')}`,
      currentProfit,
      previousProfit,
    });
  }

  if (period === 'year') {
    const yStr = String(req.query.year || '');
    if (!/^\d{4}$/.test(yStr)) {
      return res.status(400).json({ error: 'year must be YYYY' });
    }

    const year = Number(yStr);
    const prevYear = year - 1;

    const sql = `
      SELECT
        MONTH(month) AS m,
        profit
      FROM monthly_profit
      WHERE YEAR(month) = ?
      ORDER BY month
    `;

    const [curRows] = await pool.query(sql, [year]);
    const [prevRows] = await pool.query(sql, [prevYear]);

    const labels = Array.from({ length: 12 }, (_, i) => i + 1);
    const currentProfit = Array(12).fill(null);
    const previousProfit = Array(12).fill(null);

    curRows.forEach(r => { currentProfit[(r.m || 0) - 1] = Number(r.profit) || 0; });
    prevRows.forEach(r => { previousProfit[(r.m || 0) - 1] = Number(r.profit) || 0; });

    return res.json({
      labels,
      currentLabel: String(year),
      previousLabel: String(prevYear),
      currentProfit,
      previousProfit,
    });
  }

  return res.status(400).json({ error: 'period must be month|year' });
} catch (err) {
  console.error('/api/compare error:', err);
  return res.status(500).json({ error: 'Помилка сервера' });
```

```
  }
});

// --------------------
app.listen(PORT, () => {
  console.log(`Server: http://localhost:${PORT}`);
});
```

make-hash.js:

```
const bcrypt = require('bcryptjs');

async function main() {
  const password = process.argv[2];
  if (!password) {
    console.log('Вкажіть пароль: node make-hash.js MyPassword');
    process.exit(1);
  }

  const hash = await bcrypt.hash(password, 10);
  console.log('HASH:', hash);
}

main().catch(console.error);
```

package.json:

```
{
  "name": "ayvengo.analytics.app",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "type": "commonjs",
  "scripts": {
    "start": "node index.js",
    "make-hash": "node make-hash.js"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "bcryptjs": "^3.0.3",
    "body-parser": "^1.20.3",
    "cookie-parser": "^1.4.7",
    "dotenv": "^17.2.3",
    "express": "^4.22.1",
    "mysql2": "^3.15.3",
    "path": "^0.12.7"
  }
}
```