**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**
**ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОДЕСЬКОГО НАЦІОНАЛЬНОГО ТЕХНОЛОГІЧНОГО УНІВЕРСИТЕТУ»**

***Спеціальність: 121 «Інженерія програмного забезпечення»***

***Освітня програма: «Розробка програмного забезпечення»***

***Група: 4РП-05***

# Дипломний проект

**здобувача освіти денної форми навчання**
**РП.05.25.000.ДП**

***ХЛЮСТІН***
***ЯРОСЛАВ***
***ОЛЕКСІЙОВИЧ***

**м. Одеса**
**2022 р.**

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОДЕСЬКОГО НАЦІОНАЛЬНОГО ТЕХНОЛОГІЧНОГО УНІВЕРСИТЕТУ»

Спеціальність: **121 «Інженерія програмного забезпечення»**
Освітня програма: «**Розробка програмного забезпечення»**
Група: **4РП-05**

# ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проекту (роботи) на тему:

**Розробка десктопного програмного забезпечення для системи розподілу педагогічного навантаження за рахунок коштів платформи .NET**

Проектний матеріал складається з пояснювальної записки на 45 сторінках та графічного (презентаційного) матеріалу на 10 аркушах (слайдах).

Дипломник _____ (Хлюстін Я. О.)
Керівник _____ (Сологуб К.В.)

**Консультанти:**
з економічної частини _____ (Копайгородська Т.Г. )
з охорони праці_____ ( Чорновол Н.І. )
з дотримання вимог ЄСКД _____ ( Петрашова В.І.)
старший консультант _____ ( Скорнякова О.В. )

**До захисту допущений**
Голова циклової комісії _____ ( Скорнякова О.В. )
Завідувач відділення _____ (Суліма Ю.Ю.)

Захист «____»_____ 2022 р. Протокол ДКК №_____
Оцінка ДКК _____

Секретар ДКК _____

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**
**ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОДЕСЬКОГО НАЦІОНАЛЬНОГО ТЕХНОЛОГІЧНОГО УНІВЕРСИТЕТУ»**

Відділення **комп'ютерних систем** Комісія **КТ та ІІІ**

Спеціальність **121 «Інженерія програмного забезпечення»**

Освітня програма **«Розробка програмного забезпечення»**

ЗАТВЕРДЖУЮ:

Заст. дир. з НВР_____

_____

"_____"_____2022 р.

# З А В Д А Н Н Я

**на дипломний проект (роботу)**

Здобувачеві (здобувачці) освіти Хлюстін Ярослав Олексійович

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) Розробка десктопного програмного забезпечення для системи розподілу педагогічного навантаження за рахунок коштів платформи .NET

затверджена наказом по коледжу від "30" грудня 2021 р. № 306-А2-ОД

2. Термін здачі закінченого проекту (роботи) _____

3. Вихідні данні до проекту (роботи) Microsoft Visual Studio, .Net, C#, MS SQL Server, HTML CSS, JavaScript, LINQ, API, UI, UX, HTTP, Microsoft Edge, Postman

4. Зміст розрахунково-пояснювальної записки (перелік питань, які необхідно розробити)

1. Розробка системи для розподілу педагогічного навантаження з використанням asp.net. 2. Економічний розрахунок. 3. Охорона праці.

5. Перелік графічного (презентаційного) матеріалу (з точним зазначенням обов'язкових креслень, кількості слайдів)

Презентація (10 слайдів)

6. Консультанти по проекту (роботі), із зазначенням розділів проекту, що їх стосується

| Розділ | Консультант | Підпис, дата | |
|---|---|---|---|
| | | Завдання видав | Завдання прийняв |
| Основний | Сологуб К.В. | | |
| Економічний | Копайгородська Т.Г. | | |
| Охорона праці | Чорновол Н.І. | | |
| Нормоконтроль | Петрашова В.І. | | |
| Старший консультант | Скорнякова О.В. | | |

7. Дата видачі завдання _____

Керівник _____

(підпис)

Завдання прийняв до виконання _____

(підпис)

КАЛЕНДАРНИЙ ПЛАН

| № з/р | Назва етапів дипломного проекту (роботи) | Термін виконання етапів дипломного проекту (роботи) | Відмітка про виконання |
|---|---|---|---|
| 1 | Розділ 1. Розробка системи для розподілу педагогічного | | |
| | навантаження з використанням Asp.net | | |
| 2 | Розділ 2. Економічний розрахунок | | |
| 3 | Розділ 3. Охорона праці | | |
| 4 | Розробка презентації до дипломної роботи | | |
| 5 | Чистове оформлення пояснювальної записки | | |
| 6 | Підготовка доповіді до захисту | | |
| 7 | Отримання рецензії, відповіді на зауваження | | |
| | рецензента | | |
| 8 | Захист роботи | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

Дипломник _____

(підпис)

Керівник _____

(підпис)

| Формат | Зона | Поз. | Позначення | Назва | Кіл. | Примітка |
|---|---|---|---|---|---|---|
| | | | | | | |
| | | | | *Документація* | | |
| | | | | | | |
| | | | РП 05.25.000 ДП ПЗ | Дипломний проект | | |
| | | | | | | |
| А4 | | | РП 05.25.000 ДП ПЗ | Пояснювальна | 1 | |
| | | | | записка | | |
| | | | | | | |
| | | | | Презентація | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | | РП 05.25.000 | |
| | | № докум. | | | | |
| Розробив | | Хлюстін Я.О. | | | Побудова серверної частини для інтернет-магазину з використанням технологій мікросервісної архітектури | ВСП«ОТФК ОНТУ» гр.РП-05 |
| Перевірив | | Сологуб К.В. | | | | |
| | | | | | | |
| Н. Контр. | | Петрашова В.І | | | | |
| Затверд. | | Скорнякова О.В. | | | | |

# АНОТАЦІЯ

**Об'єкт дослідження:** об'єктом дослідження є процес розробки платформи для освітнього закладу

**Предмет дослідження:** предметом дослідження стала система для розподілу педагогічного навантаження.

**Мета роботи:** Розробка рекламного веб-сайту для просування компанії та пошуку нових клієнтів.

**Досягнуті результати:**

- Вибрані необхідні засоби розробки;
- Розроблено архітектуру взаємодії з модулем "Педагогічна навантаження"
- Розроблено архітектуру пакетів системи;
- Створено необхідні таблиці у базі даних для модуля "Педагогічна навантаження";
- Виявлено проблеми непрацюючого та незатребуваного функціоналу та враховано у розробленому модулі "Педагогічна навантаження"

**Ключові слова:** HTML, CSS, JavaScript, Bootstrap.

**Обсяг:** 45 стор., 18 Рисунок, 13 табл., 8 джерел.

**ЗМІСТ**

# ВСТУП

Найважливіше завдання сфери освіти – задоволення освітніх потреб суспільства загалом і кожного учня зокрема. З розвитком інформаційних технологій та реформуванням вищої освіти виникають потреби у перегляді підходу до забезпечення якості освіти. Однак, коли йдеться про якість освіти, часто мається на увазі якість результатів діяльності освітньої установи, що призводить до думки, що ефективність роботи навчального закладу безпосередньо впливає на ефективність навчання студентів. У свою чергу ефективність роботи навчального закладу залежить від роботи професорсько-викладацького складу. На працездатність останніх впливає якісний розподіл навчального навантаження.

Під навчальним навантаженням викладачів вищих навчальних закладів розуміється загальна кількість годин оплачуваної викладацької роботи у навчальний рік, яка виконується ними в одній милі кількох освітніх закладах.

Алгоритм розподілу навчального навантаження у ВНЗ досить трудомісткий і схильний до помилок процес, в якому людині, яка займається цим, необхідно знати всі нормативні документи. Це супроводжує великий обсяг «паперової» роботи. Автоматизація виконання може допомогти уникнути помилок, зумовлених людським фактором, а процес розподілу навантаження буде спрощений і прискорений.

Мета роботи – здійснити часткове перенесення функціоналу на веб-платформу для розподілу педагогічного навантаження.

Для досягнення мети необхідно вирішити такі завдання:

- Спроектувати архітектуру платформи, що складається з підмодулів: «Викладачі», «Навчальне навантаження»
- розробити та ввести в дослідну експлуатацію на кафедрі «КТ та ПІ» Одеського технічного професійного коледжу

# 1 РОЗРОБКА СИСТЕМИ ДЛЯ РОЗПОДІЛУ ПЕДАГОГІЧНОГО НАВАНТАЖЕННЯ З ВИКОРИСТАННЯМ ASP.NET

## 1.1 Проектування архітектури

Архітектура програми містить щонайменше один проект. У такому разі вся логіка програми укладена в одному проекті, компілюється в одну збірку та розгортається як один елемент.

Будь-який створюваний у Visual Studio або з командного рядка проект ASP.NET Core спочатку буде комплексним монолітним проектом. У ньому буде укладено всю поведінку програми, включаючи презентацію даних, бізнес-логіку та логіку доступу до даних. На рис. 1.1 показано файлову структуру програми, що складається з одного проекту.

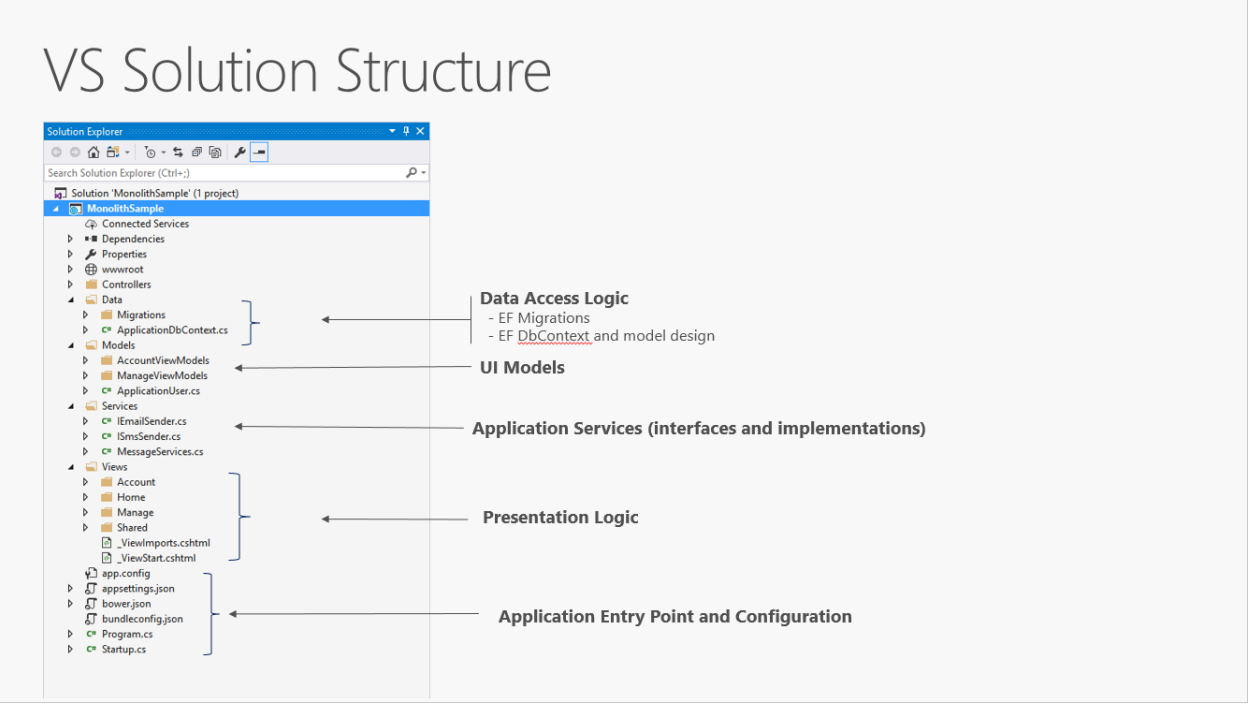


**Рисунок 1.1 – Структура проекту**

У сценарії з одним проектом поділ завдань реалізується за допомогою папок. За замовчуванням шаблон включає окремі папки для обов'язків шаблону MVC (моделі, уявлення та контролери), а також додаткові папки для

даних і служб. При такій організації деталі презентації даних максимально можливо розміщуються в папці уявлень (Views), а деталі реалізації доступу до даних повинні бути обмежені класами, що містяться в папці даних (Data). Бізнес-логіка при цьому розміщується у службах та класах, що знаходяться у папці моделей (Models).

Незважаючи на свою простоту, монолітне рішення з одним проектом має певні вади. У міру збільшення розміру та складності проекту зростатиме кількість файлів та папок. Завдання, пов'язані з інтерфейсом користувача (моделі, уявлення, контролери), розміщуються в різних папках, які не впорядковані за абеткою. З додаванням в окремі папки конструкцій рівня інтерфейсу користувача, наприклад фільтрів або зв'язувачів моделі, ситуація тільки погіршується. Бізнес-логіка втрачається в папках моделей (Models) і служб (Services), у результаті неможливо чітко визначити, які класи у яких папках мають залежати від інших класів. Подібна неефективна організація на рівні проекту часто призводить до отримання погано структурованого коду.

Для вирішення подібних проблем програми часто організуються як рішення, що складаються з безлічі проектів, де кожен проект розміщується в окремому шарі програми.

У міру збільшення складності програми для ефективного управління ним може застосовуватися розбиття за обов'язками та завданнями. Такий підхід відповідає принципу поділу завдань і допомагає зберегти організацію бази коду, що розширюється, завдяки чому розробники можуть швидко визначати, де саме реалізовані певні функції. Багатошарова архітектура має також низку інших переваг.

Завдяки впорядкуванню коду за допомогою шарів, загальні низькорівневі функції можуть багаторазово використовуватися по всьому додатку. Це дуже важливо, оскільки такий підхід вимагає меншого обсягу коду і, за рахунок стандартизації програми на рівні однієї реалізації, відповідає принципу "Не повторюйся".

У програмах з багатошаровою архітектурою можуть встановлюватися обмеження взаємодії між шарами. Така архітектура допомагає продати інкапсуляцію. При зміні або заміні шару будуть порушені ті шари, які працюють безпосередньо з ним. Обмежуючи залежності шарів один від одного, можна зменшити наслідки внесення змін, внаслідок чого одинична зміна не впливатиме на всю програму.

Застосування шарів (і інкапсуляція) дозволяє помітно спростити заміну функціональних можливостей у межах програми. Наприклад, програма може спочатку використовувати власну базу даних SQL Server для збереження, а згодом перейти на стратегію збереження стану на основі хмари або веб-API. Якщо у програмі належним чином інкапсульована реалізація збереження на логічному шарі, цей шар SQL Server може бути замінений новим, де буде реалізовуватися той самий відкритий інтерфейс.

Крім можливості заміни реалізацій у зв'язку з наступними змінами, застосування шарів у додатку також дозволяє змінювати реалізації з метою тестування. Замість написання тестів, які застосовуються до шарів реальних даних або інтерфейсу програми, під час тестування вони замінюються фіктивними реалізаціями, які демонструють відому реакцію на запити. Як правило, це значно спрощує написання тестів та прискорює їх виконання порівняно з тестуванням у реальній інфраструктурі програми.

Поділ на логічні шари поширений і допомагає впорядкувати код додатків підприємства. Зробити це можна кількома способами. Загальноприйнята організація логіки програми по шарах показано на рис. 1.2.

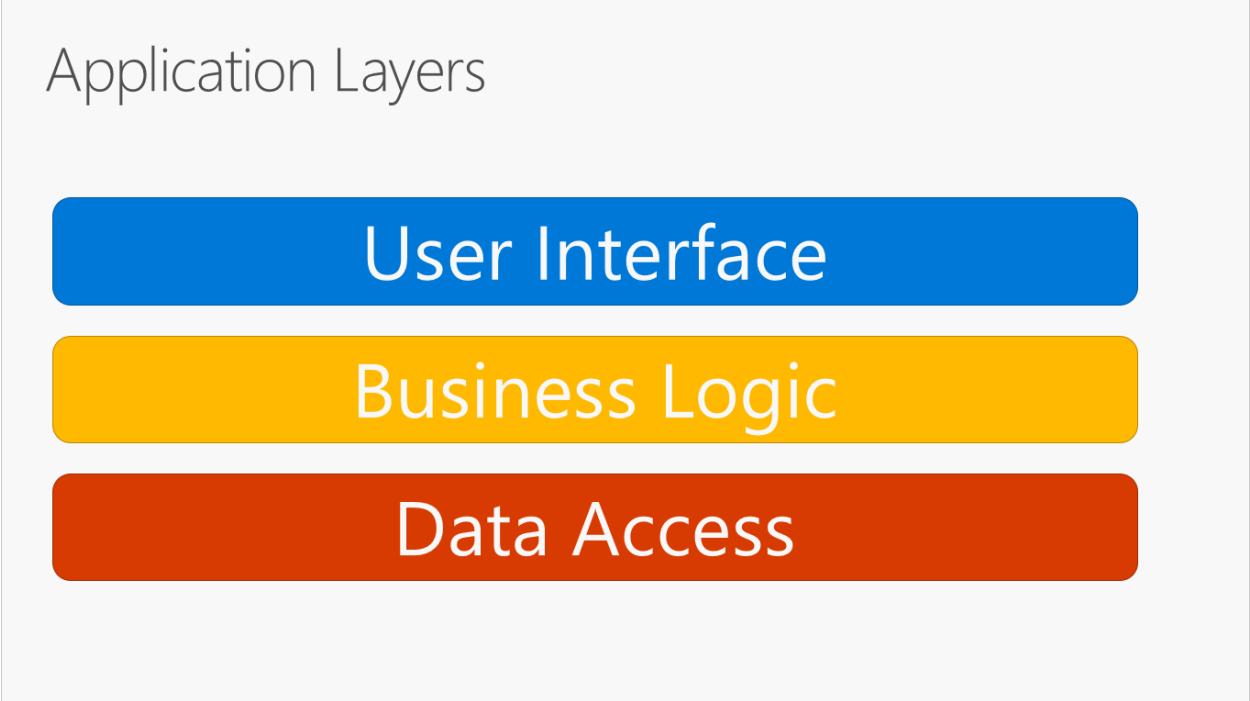


**Рисунок 1.2 – Шари додатку**

Як правило, у додатку визначаються шари інтерфейсу користувача, бізнес-логіки і доступу до даних. У рамках такої архітектури користувачі виконують запити через шар інтерфейсу користувача, який взаємодіє тільки з шаром бізнес-логіки. Шар бізнес-логіки, у свою чергу, може спричинити шар доступу до даних для обробки запитів. Шар інтерфейсу користувача не повинен виконувати запити безпосередньо до шару доступу до даних та будь-якими іншими способами безпосередньо взаємодіяти з функціями збереження. Аналогічним чином, шар бізнес-логіки повинен взаємодіяти з функціями збереження лише через шар доступу до даних. Таким чином, для кожного шару чітко визначено свій обов'язок.

Одним із недоліків традиційного багатошарового підходу є те, що обробка залежностей під час компіляції здійснюється зверху донизу. Це означає, що шар інтерфейсу користувача залежить від шару бізнес-логіки, який, у свою чергу, залежить від шару доступу до даних. Це означає, що шар бізнес-логіки, який зазвичай містить ключові функції програми, залежить від деталей реалізації доступу до даних (і найчастіше від самої бази даних). Тестування бізнес-логіки в такій архітектурі найчастіше утруднене і потребує

тестової бази даних. Для вирішення цієї проблеми можна застосовувати принцип інверсії залежностей, як описано в наступному розділі.

На рис. 1.3 показаний приклад рішення, в якому додаток поділено на три проекти (або шари) відповідно до певних обов'язків.

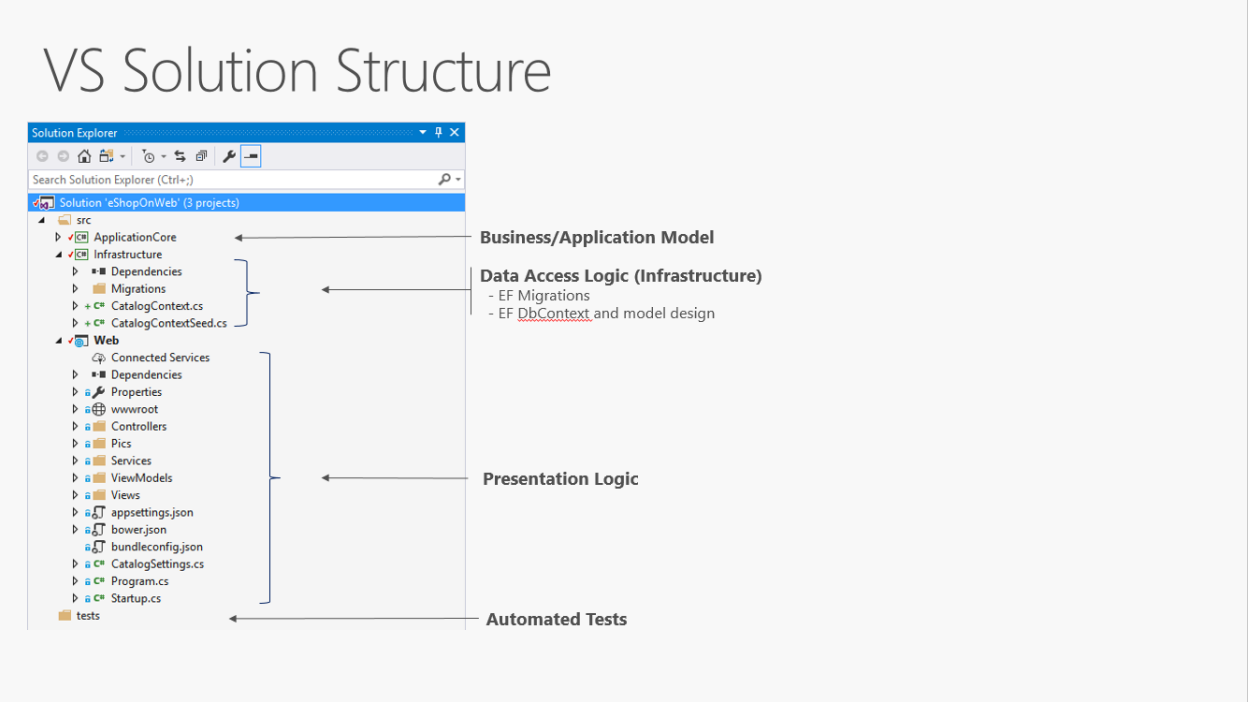


**Рисунок 1.3 – Простий монолітний додаток, що складається з трьох проектів.**

Незважаючи на те, що в цілях упорядкування в цьому додатку використовується кілька проектів, воно, як і раніше, розгортається як єдиний елемент, і його клієнти взаємодіють з ним як з одним веб-додатком. Це дозволяє реалізувати вкрай простий процес розгортання. На рис. 1.4 показано, як таку програму можна розмістити з використанням Azure.

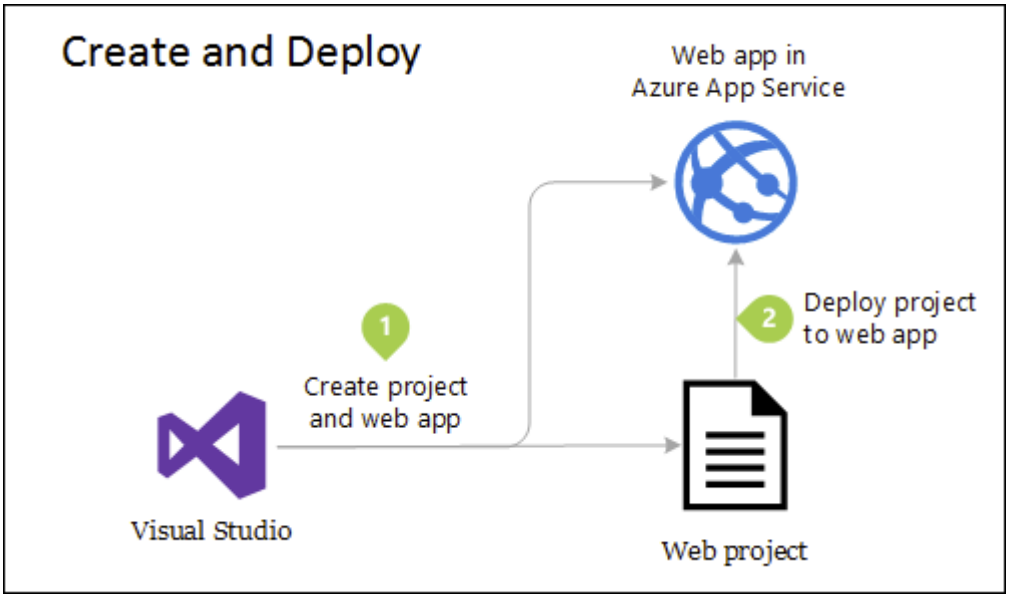


**Рисунок 1.4 -Просте розгортання веб-програми Azure**

У міру розвитку програми можуть знадобитися більш складні та надійні рішення для розгортання. На рис. 1.5 показаний приклад складнішого плану розгортання, який підтримує додаткові можливості.

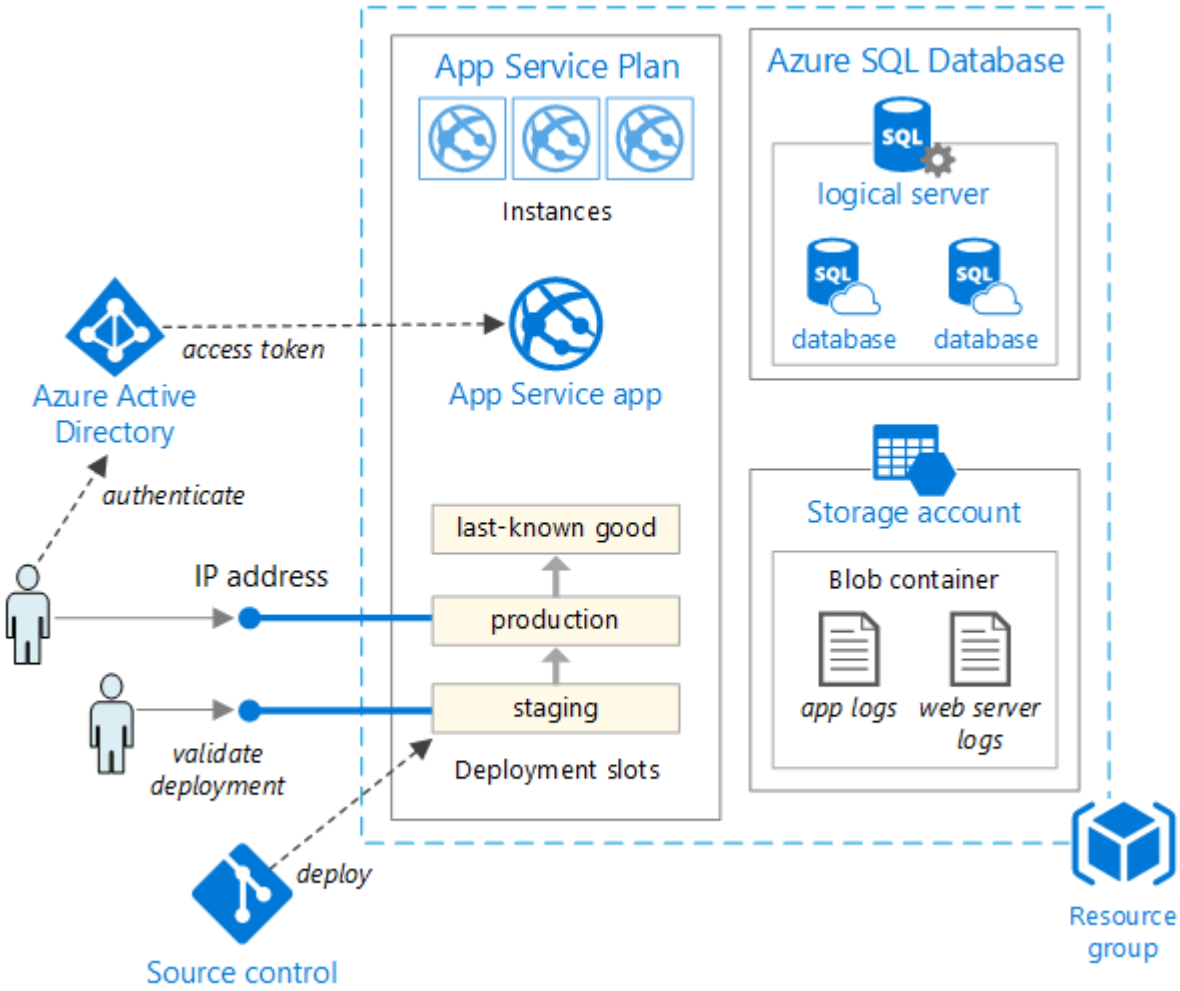


**Рисунок 1.5 - Розгортання веб-програм у службі додатків Azure**

Розбиття цього проекту на кілька проектів на основі обов'язків дозволяє підвищити зручність підтримки програми.

Такий елемент підтримує вертикальне та горизонтальне масштабування, що дозволяє використовувати переваги хмарного масштабування на запит. Під вертикальним масштабуванням розуміється збільшення кількості ЦП, обсягу пам'яті, місця на диску та інших ресурсів на серверах, де розміщується програма. Горизонтальне масштабування полягає у додаванні додаткових екземплярів таких фізичних серверів, віртуальних машин чи контейнерів. Якщо програма розміщується на кількох примірниках, для розподілу запитів між примірниками програми використовується система балансування навантаження.

Найпростіший підхід до масштабування веб-програми в Azure полягає в ручному налаштуванні масштабування в плані служби програм для програми. На рис. 5-6 показаний екран панелі моніторингу Azure, призначений для налаштування числа екземплярів, що обслуговують програму.

Рішення з чистою архітектурою для кожного проекту чітко визначено обов'язки. Фактично, кожному проекту належать певні типи, а проекти будуть представлені відповідні цим типам папки.

Ядро програми містить бізнес-модель, яка включає сутності, служби та інтерфейси. Такі інтерфейси включають абстракції для операцій, які будуть виконуватися з використанням архітектури, включаючи операції доступу до даних або файлової системи, мережні дзвінки тощо. У деяких випадках служби або інтерфейси, визначені в цьому шарі, повинні працювати з типами, що не є типами сутностей, які не мають залежностей від інтерфейсу користувача або інфраструктури. Вони можуть визначатися як найпростіші об'єкти передачі даних.

Типи ядра програми

- Сутності (класи бізнес-моделі, що зберігаються)
- Агрегати (групи сутності).
- Інтерфейси
- Доменні служби
- Специфікації
- Користувальницькі винятки та пропозиції умов.
- Події домену та обробники.
- Інфраструктура

Як правило, проект інфраструктури включає в себе реалізацію доступу до даних. У типовій веб-програмі ASP.NET Core ця реалізація включає Entity Framework (EF) DbContext, будь-які об'єкти Migration EF Core, а також класи реалізації доступу до даних. Найбільш поширений підхід до абстрагування

коду реалізації доступу до даних полягає у використанні конструктивного шаблону репозиторію.

Крім реалізації доступу до даних, проект інфраструктури також повинен включати реалізації служб, які повинні взаємодіяти з інфраструктурними завданнями. Ці служби повинні здійснювати інтерфейси, визначені в ядрі програми. Таким чином, інфраструктура повинна містити посилання на проект ядра програми.

- Типи інфраструктури
- Типи EF Core (DbContext, Migration)
- Типи реалізації доступу до даних (репозиторії)
- Служби, пов'язані з інфраструктурою (наприклад, FileLogger або SmtpNotifier)
- Рівень інтерфейсу користувача

Шар інтерфейсу користувача в програмі MVC ASP.NET Core виступає як точка входу для програми. Цей проект повинен посилатися на шар ядра програми, яке типи повинні взаємодіяти з інфраструктурою суворо через інтерфейси, визначені в ядрі програми. У шарі інтерфейсу користувача не повинні дозволятися пряме створення екземплярів для типів шару інфраструктури, а також їх статичні виклики.

Типи шарів інтерфейсу користувача

- Контролери
- Фільтри, що настроюються.
- Користувальницьке програмне забезпечення проміжного шару.
- Уявлення
- Моделі уявлень
- Запуск

Клас Startup або файл Program.cs відповідає за налаштування програми та зв'язування типів реалізації з інтерфейсами. Розташування, де виконується ця логіка, називається коренем композиції програми. Він забезпечує правильне використання залежностей під час виконання.

### 1.2 Проектування бази даних

Для зберігання даних потрібна база даних. Ми можемо використовувати різні СУБД, але, як правило, як база даних у зв'язці з ASP.NET MVC використовується база даних MS SQL Server, на прикладі якого ми і подивимося весь процес створення БД та підключення до неї.

Ми можемо створити базу даних у проекті, або ж створити її на сервері MS SQL. Для зберігання баз даних у проекті призначено папку App_Data. Отже, натиснемо на папку App_Data правою кнопкою миші і в контекстному меню виберемо Add-> New Item .... У вікні додавання нового елемента виберемо SQL Server Database і назвемо нову базу даних otfk.mdf:

Після цього до папки App_Data буде додано базу даних, і ми можемо починати з нею працювати - додавати таблиці та дані. Але перед цим подивимося, що ми будемо зберігати в БД.

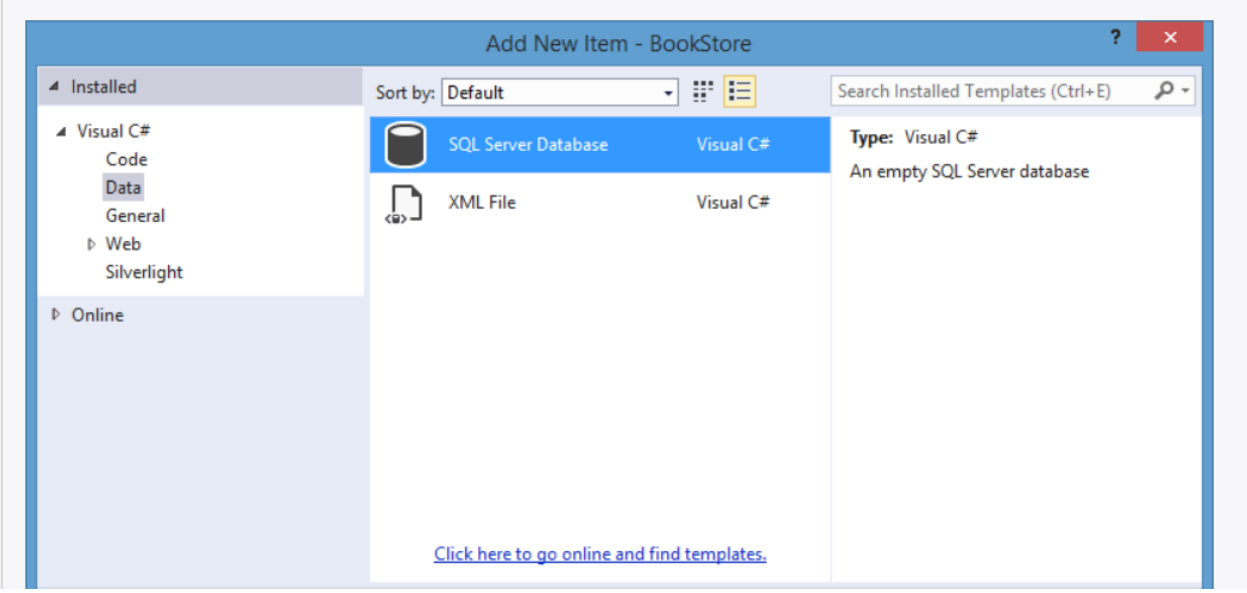


**Рисунок 1.6 – Створення бази даних**

Щоб зв'язати програму, контекст даних та БД, додамо у файл web.config рядок підключення до цієї бази даних. Для цього після секції configSections вставимо наступну секцію:

```
<connectionStrings>
    <add name="BookContext" connectionString="Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename='|DataDirectory|\otfk.mdf';Int
egrated Security=True"
 providerName="System.Data.SqlClient"/>
  </connectionStrings>
```

Архітектура бази даних була реалізована і використовувалася представлена схема малюнку 1.8.

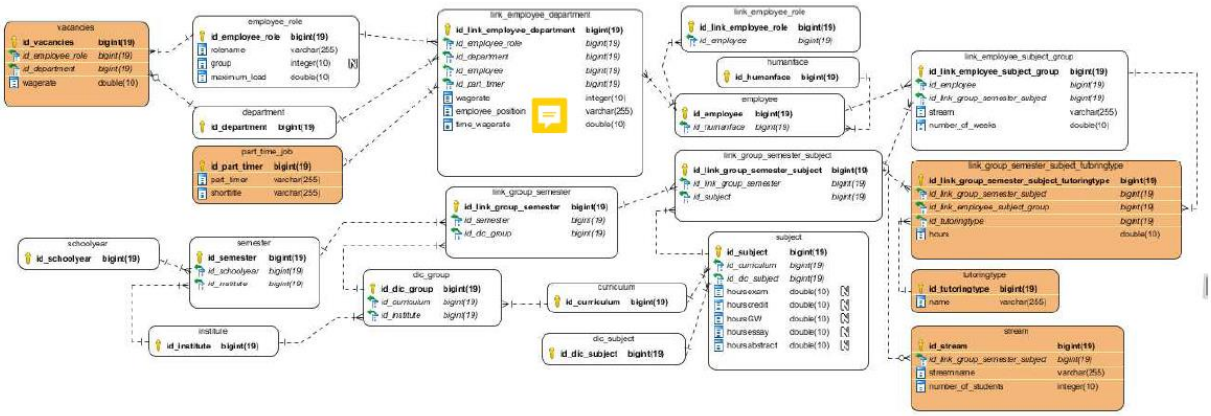

**Рисунок 1.7 – Схема БД для проекту**

Для зберігання та вилучення даних ми будемо використовувати технологію доступу до даних .NET Framework, відому як Entity Framework, для визначення та роботи з моделями.

Entity Framework (EF) підтримує метод Code First, який дозволяє створювати об'єкти моделі шляхом написання простих класів, а потім база даних буде створюватися на льоту з класів, що забезпечує дуже чистий і швидкий робочий процес розробки.

### 1.3 Проектування UI для системи

Фронтенд-розробка - область веб-розробки, що фокусується на взаємодії з користувачем ресурсу, за рахунок перетворення коду на зовнішній графічний

інтерфейс браузера, що спрощує сприйняття та розуміння представленої на сайті інформації.

Саме дизайн зовнішнього інтерфейсу приваблює аудиторію на ваш сайт і утримує її там протягом тривалого часу, роблячи його зручним для читання, привабливим і легким для сприйняття. Загалом, все, що ви бачите на екрані браузера: текст, картинки, кнопки, меню і панель навігації, що з'являється з нізвідки, - справа рук фронтенд-розробника.

Цей напрямок бурхливо розвивається і має величезну спільноту послідовників, безліч готових рішень для будь-якого проекту та тисячі кілометрів написаного коду, тому його варто розглянути як один із можливих варіантів майбутньої IT-професії.

Заради справедливості необхідно згадати ще й бекенд-розробку, що відповідає за взаємодію ресурсу з сервером, але це вже зовсім інша історія.

Інтерфейсні фахівці використовують при роботі безліч технологій, однак незмінними, фундаментальними для цієї сфери інструментами є: мова гіпертекстової розмітки (HTML), каскадні таблиці стилів (CSS) та мова програмування JavaScript.

Розберемо кожен із них докладніше:

HTML

За допомогою HTML ми розмічаємо сторінку. Ділимо її на блоки та наповнюємо їх необхідним контентом: текстом, зображеннями та відео. Будівельні елементи мови - це теги - ключові слова, що дають браузеру зрозуміти, що це за елемент і як його правильно відобразити. Більшість тегів складаються з двох частин: що відкриває та закриває, між якими пишеться необхідний нам контент, наприклад:

 // тег, що відкриває

// якийсь контент

 // тег, що закриває

CSS

CSS (каскадна таблиця стилів) — набір параметрів, які застосовуються до HTML-елементу керувати його зовнішнім виглядом. Основна роль стильового оформлення зробити контент привабливішим читача.

Справа в тому, що всі HTML-теги за замовчуванням мають нічим не примітний вигляд. Якщо їх грамотно не виділити на сторінці, мозок користувача може неправильно інтерпретувати надану інформацію.

За допомогою CSS можна налаштувати безліч різних параметрів: розміри, колір, прозорість, розташування блоків по осі, тінь, світіння, зміна при наведенні, градієнт, обведення та багато іншого.

JavaScript:

JavaScript (JS) — скриптова мова для браузерів, призначена для перехоплення подій та виконання різних дій.

Припустимо, один із користувачів вашого ресурсу клікнув кнопкою миші на якійсь ділянці веб-сторінки, тим самим викликавши подію click. За допомогою JS ми можемо пов'язати цю подію з виконанням потрібної функції, наприклад, викликати відкриття модального вікна або зміна кольору елемента.

Підказки, що обертаються, зображення, що обертаються, різного роду падаючі сніжинки та інші анімешки - все це робота JS.

Розробники використовують мову, щоб зробити сайт інтерактивним, тобто дати користувачеві можливість взаємодіяти з його елементами. Чуйність ресурсу робить його цікавішим. JavaScript люблять і ненавидять, але без нього в сучасній веб-розробці не обійтися!

Зайти в інтернет-браузер тепер можна з будь-якого пристрою, тому розробник повинен адаптувати свій проект до різних пристроїв з різними розмірами/параметрами екранів. Ви ж не хочете, щоб ваш сайт поплив убік чи навпаки — всі елементи наїхали один на одного. Знання принципів адаптивного дизайну та способів його впровадження у код має вирішальне значення у сучасній верстці. Верстальник - це невід'ємна іпостась фронтендера.

Також необхідно враховувати поведінку вашого ресурсу у популярних серед користувачів браузерах. Хоча більшість із них в даний час здатні підтримувати однаковість при відображенні вмісту сайту, все одно є ряд властивостей, для яких потрібно прописувати додаткові префікси, щоб подружити їх із браузером.

Коли обсяг написаного CSS збільшується, існує ризик ненароком перевизначити вже написані властивості, помилившись з ім'ям класу, і витратити час на написання одного і того ж коду повторно. Тут у гру вступає попередня обробка стилів з допомогою препроцессорів CSS. По суті це ще один спосіб полегшити собі життя. SASS, LESS і Stylus - рішення, що дозволяють розробникам створювати код мовою препроцесора, грамотно його структурувати, а потім перетворювати його на звичайний стильовий файл. Такий підхід дозволяє легко редагувати код та уникнути вищезгаданих помилок.

Фреймворки та бібліотеки JS

Бібліотека – це набір плагінів та розширень, що прискорюють використання JS при розробці сайту. Їхнє використання допомагає скорочувати звичайні багаторядкові операції до структури, яку можна виконати за допомогою кількох рядків коду. Їх створено чимало. Найбільшу популярність серед розробників набули такі фреймворки як React, Vue.js, Angular. Втім, життя є і поза великою трійкою.

Фреймворки мають свої переваги і недоліки, і дуже важливо зрозуміти, яка платформа ідеально підходить для проекту, що розробляється. Однак починати їх вивчення потрібно лише тоді, коли розробник досконально розібрався з усіма аспектами мови (JavaScript).

### 1.4 Проектування RestAPI

В останні роки стало зрозуміло, що HTTP не лише для обслуговування HTML-сторінок. Це також потужна платформа для створення веб-API,

використовуючи кілька команд (GET, POST тощо), а також кілька простих понять, таких як URI та заголовки. веб-API ASP.NET – це набір компонентів, що спрощують програмування HTTP. Оскільки створено на основі середовища виконання ASP.NET MVC, веб-API автоматично обробляє низькорівневі відомості про транспорт HTTP. У той же час веб-API надає модель програмування HTTP. Насправді одна з цілей веб-API полягає в тому, щоб не абстрагувати реальність HTTP. В результаті веб-API є гнучким та простим для розширення. Стиль архітектури REST виявився ефективним способом використання протоколу HTTP, хоча це, безумовно, не єдиний допустимий підхід до HTTP. Диспетчер контактів надасть RESTful для перерахування, додавання та видалення контактів, серед іншого.

Принципи REST API

RESTful має 7 принципів написання коду інтерфейсів.

Відділення клієнта від сервера (Client-Server). Клієнт — це інтерфейс сайту або програми, наприклад, пошуковий рядок відеохостингу. У REST API код запитів залишається на стороні клієнта, а код доступу до даних - на стороні сервера. Це спрощує організацію API, дозволяє легко переносити інтерфейс користувача на іншу платформу і дає можливість краще масштабувати серверне зберігання даних.

Відсутність запису стану клієнта (Stateless). Сервер не повинен зберігати інформацію про стан (проведені операції) клієнта. Кожен запит від клієнта повинен містити тільки інформацію, яка потрібна для отримання даних від сервера.

Кешування (Casheable). У даних запиту має бути зазначено, чи потрібно кешувати дані (зберігати у спеціальному буфері для частих запитів). Якщо така вказівка є, клієнт отримає право звертатися до цього буфера за потреби.

Єдність інтерфейсу (Uniform Interface). Всі дані повинні запитуватись через одну URL-адресу стандартними протоколами, наприклад, HTTP. Це спрощує архітектуру сайту або програми та робить взаємодію з сервером зрозумілішим.

Багаторівневість системи (Layered System). У RESTful сервері можуть розташовуватися різних рівнях, у своїй кожен сервер взаємодіє лише з найближчими рівнями і пов'язаний запитами коїться з іншими.

Надання коду на запит (Code on Demand). Сервери можуть надсилати клієнтові код (наприклад, скрипт для запуску відео). Так загальний код програми або сайту стає складнішим лише за необхідності.

Початок від нуля (Starting with the Null Style). Клієнт знає лише одну точку входу на сервер. Подальші можливості взаємодії забезпечуються сервером.

Сам собою RESTful не є стандартом або протоколом. Розробники керуються принципами REST API для створення ефективної роботи з серверами для своїх сайтів та додатків. Принципи дозволяють будувати серверну архітектуру за допомогою інших протоколів: HTTP, URL, JSON та XML.

Це відрізняє REST API від методу простого протоколу доступу до об'єктів SOAP (Simple Object Access Protocol), створеного Microsoft у 1998 році. У SOAP взаємодію кожного протоколу потрібно прописувати окремо лише форматі XML. Також у SOAP немає кешування запитів, більш об'ємна документація та реалізація словника, окремого від HTTP. Це робить стиль REST API більш легким у реалізації, ніж стандарт SOAP.

Незважаючи на відсутність стандартів, при створенні REST API є загальноприйняті найкращі практики, наприклад:

- використання захищеного протоколу HTTPS
- використання інструментів для розробки API Blueprint та Swagger
- застосування програми для тестування Get Postman
- застосування якомога більшої кількості HTTP-кодів (список)
- архівування великих блоків даних

У REST API є 4 методи HTTP, які використовують для дій з об'єктами на серверах:

- GET (отримання інформації про дані або список об'єктів)

- DELETE (видалення даних)
- POST (додавання або заміна даних)
- PUT (регулярне оновлення даних)

Такі запити ще називають ідентифікаторами CRUD: create (створити), read (прочитати), update (оновити) delete (видалити). Це стандартний набір дій для роботи із даними.

У кожному HTTP-запиті є заголовок, за яким слідує опис об'єкта на сервері - це і є його стан.

## 1.5 Робота з платформою

### 1.5.1 Опис сторінки «Викладачі»

Даний підмодуль дозволяє працювати з викладачами на певній цикловій комісії, тобто додавати їх до комісії, прибирати їх з комісії, а також визначати їм наступні параметри:

- Інформацію про сумісництво
- Посада
- Величина ставки
- Величина норми часу на ставку

Також підмодуль дозволяє переглядати у викладача максимально допустиме навантаження та скільки ще навантаження їм можна призначити.

Додати викладача можна двома способами. Перший спосіб – це додавання викладача із зазначенням його посади при натисканні на кнопку "Додати ставку". У спливаючому вікні, що показано рис. 2.1, спочатку необхідно здійснити пошук потрібного викладача (викладачі завантажуються з бази даних після натискання на кнопку "Пошук"), вибрати його, призначити йому посаду і додати.

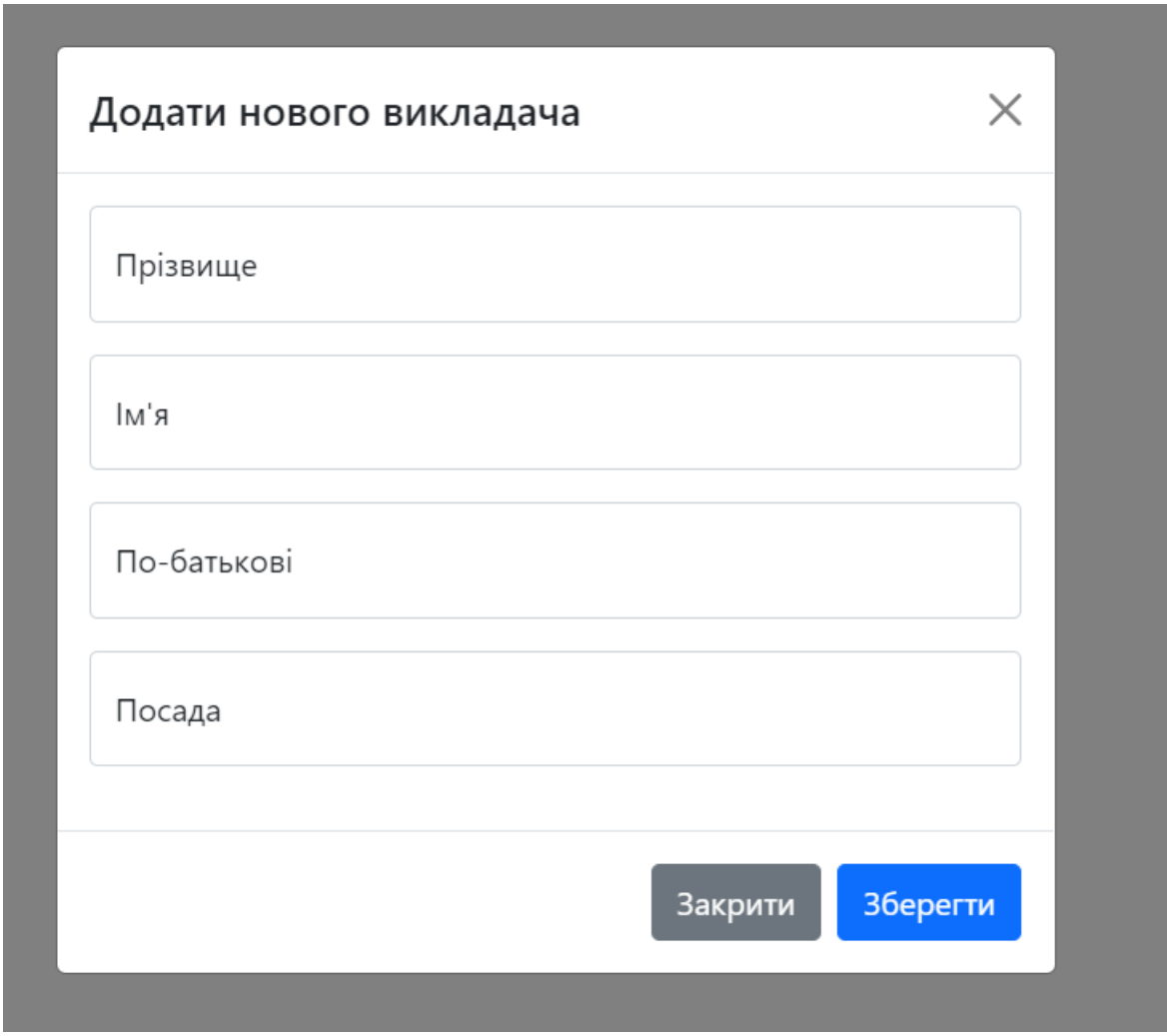


**Рисунок 1.8 – Додавання викладача**

Другий спосіб полягає у використанні вакансій. Трапляються такі випадки, коли потрібна посада, на яку треба призначити викладача, відома, а сам викладач не відомий. Отже, для зручності роботи з системою можна зберегти обрану посаду з розміром ставки у вигляді вакансії, а потім пізніше на створену вакансію призначити якогось певного викладача. Для зберігання вакансій було створено окрему таблицю у базі даних. Такі можливості як додавання та зміна вакансії представлені на рисунках 1.9 та 1.10.

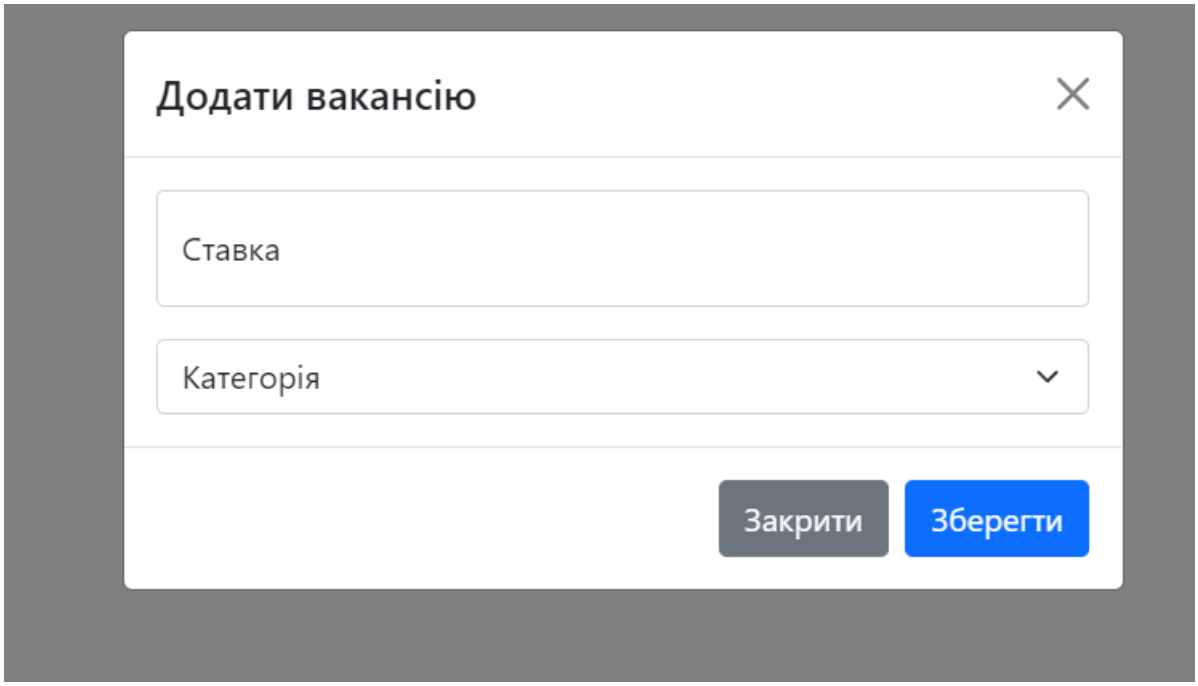


**Рисунок 1.9 – Додавання вакансії**

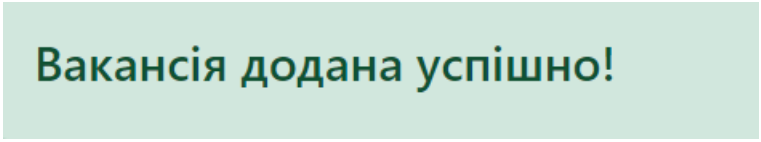


**Рисунок 1.10 – Зміна вакансії**

Також вакансії можна видаляти та, найголовніше, заповнювати. Після заповнення вакансії відбувається закріплення викладача за кафедрою із зазначеними у вакансії параметрами. Спливне вікно заповнення вакансії представлено малюнку 1.11.

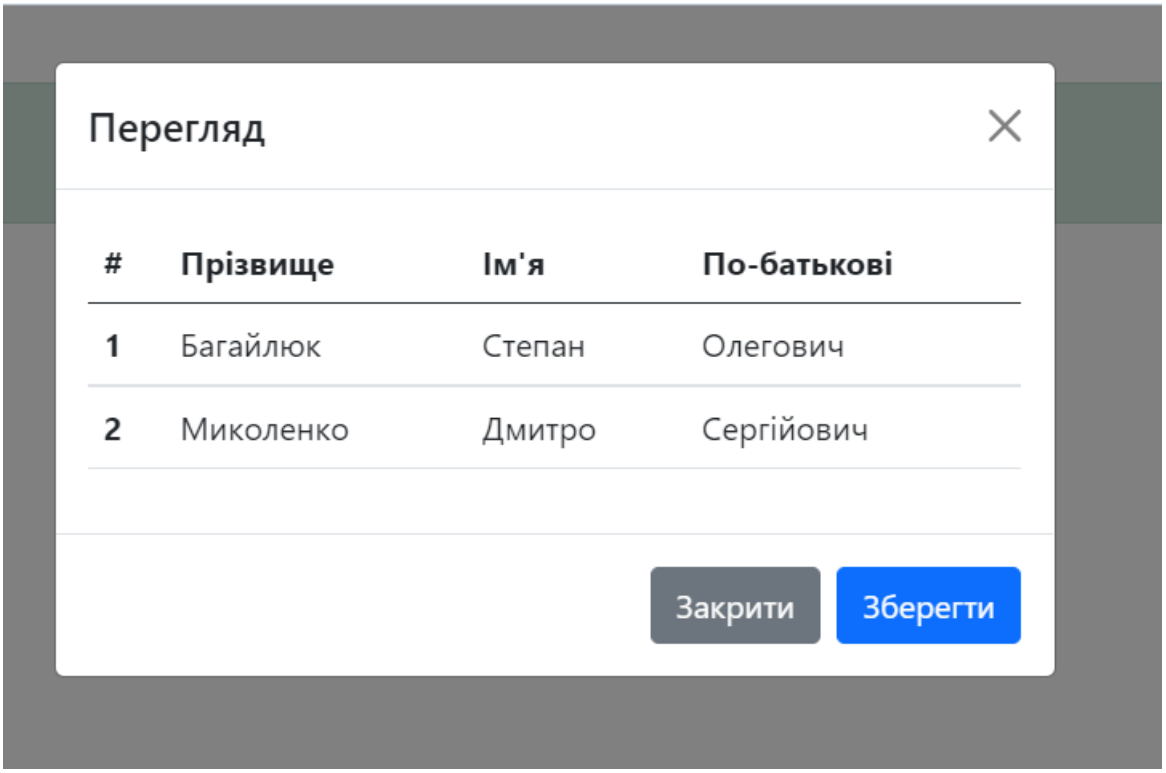


**Рисунок 1.11 – Заповнення вакансії**

Додати викладача на кафедру означає додати запис до бази даних у таблиці, де реалізовано зв'язок викладача та кафедри.

Область, де подано дані про зайнятість викладача на кафедрі, показано на рисунку 1.12.

| # | Прізвище | Ім'я | По-батькові | Ставка | Норма часу | Відхилення | Максимальне навантаження |
|---|---|---|---|---|---|---|---|
| 1 | Багайлюк | Степан | Олегович | 0.1 | 0 | 800 | 800 |
| 2 | Миколенко | Дмитро | Сергійович | 0.1 | 0 | 800 | 800 |

**Рисунок 1.12 – Інформація о викладачах**

Зберігши існуючий функціонал до підмодуля "Викладачі" було додано максимальне навантаження та відхилення за викладачем, щоб була можливість визначити, скільки годин можна призначити викладачеві, який працює на обраній у підмодулі кафедрі. У вже існуючій програмі це було неможливо (кількість годин по викладачеві була відома лише на кафедрі, яку обрали у підмодулі). Для викладачів, які поєднують роботу на кількох кафедрах, необхідно передбачити обмеження на сумарну кількість годин. Це виконують додані функції.

Максимальне навантаження – це загальна кількість годин на посаду. Відхилення - це кількість годин, що залишилася у викладача, з урахуванням навантаження на інших кафедрах.

Параметр "Сумісництво" може приймати такі значення як: "Зовнішній", "Внутрішній" та "Штатний". Ставка і норма часу ставку вказується користувачем, відхилення обчислюється програмно, а значення максимальної навантаження зберігається у базі даних.

### 1.5.2 Сторінка «Педагогічне навантаження»

Підмодуль "Педагогічна навантаження" для ОТФК дозволяє:

- автоматизувати процес розподілу навчального навантаження серед викладачів кафедри
- здійснювати допоміжні процеси, пов'язані з розподілом навантаження, такі як генерація звітів, планів викладача та доручень.

У підмодулі "Педагогічний навантаження" користувач має можливості:

- переглянути таблицю "Педагогічне навантаження" з детальною інформацією щодо типів контролю на дисципліну за обраною кафедрою;
- закріпити викладача за певним типом контролю за дисципліною;
- відкріпити викладача від обраного типу контролю з дисципліни;
- зібрати рядок;
- розбити рядок;
- вибрати колонки, що переглядаються;
- експортувати дані у форматах .pdf та .xls.

Скріншоти роботи закріплення та відкріплення типів контролю з дисципліни представлені на рисунках 1.13

| # | Прізвище | Ім'я | По-батькові | Ставка | Норма часу | Відхилення | Максимальне навантаження | Кількість студентів | Вид заняття | Аудиторная | Загальна |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Багайлюк | Степан | Олегович | 0.1 | 0 | 800 | 800 | 123 | Лекція | 1020 | 1050 |
| 2 | Миколенко | Дмитро | Сергійович | 0.1 | 0 | 800 | 800 | 123 | Лекція | 1020 | 1050 |

**Рисунок 1.13 – Результат закріплення викладача за дисципліною**

### 1.5.3 Робота з групами

Команда "Зібрати рядок" є функціонал для злиття підгруп в один тип контролю з дисципліни (рис. 1.14)

Основні дії необхідної послідовності:

- Затиснути клавішу Ctrl та виділити необхідні рядки типів контролю на дисципліну різних підгруп;
- Натиснути кнопку "Зібрати";
- У вікні, що відображає вибрані рядки та результат їх злиття, натиснути на кнопку "Прийняти".

| # | Прізвище | Ім'я | По-батькові | Ставка | Норма часу | Відхилення | Максимальне навантаження |
|---|---|---|---|---|---|---|---|
| 1 | Багайлюк | Степан | Олегович | 0.1 | 0 | 800 | 800 |
| 2 | Миколенко | Дмитро | Сергійович | 0.1 | 0 | 800 | 800 |

| # | Прізвище | Ім'я | По-батькові | Ставка | Норма часу | Відхилення | Максимальне навантаження | Кількість студентів | Вид заняття | Аудиторная | Загальна |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Багайлюк | Степан | Олегович | 0.1 | 0 | 800 | 800 | 123 | Лекція | 1020 | 1050 |
| 2 | Миколенко | Дмитро | Сергійович | 0.1 | 0 | 800 | 800 | 123 | Лекція | 1020 | 1050 |

Зібрати

**Рисунок 1.14 – Збирання рядків**

У підмодулі "Педагогічна навантаження", функція "збирати рядок" тепер поєднує потрібну кількість розбитих рядків. На рисунках 1.15 показано розбиття типу контролю на дисципліну "Об'єктно-орієнтоване програмування" на 3 підгрупи, за потреби ми можемо зібрати два з них, результати злиття двох рядків

| # | Дисципліна | Закріплення | Курс | Кількість студентів | Вид заняття |
|---|---|---|---|---|---|
| 1 | ООП | Багайлюк Степан Олегович | 3 | 32 | Аудиторна |
| 1 | ООП | Багайлюк Степан Олегович | 3 | 28 | Аудиторна |

**Рисунок 1.15 - Збереження**

"Розбити рядок" є функціонал для розбиття потоку на підгрупи з дисципліни.

Основні дії необхідної послідовності:

виділити необхідний рядок однієї дисципліни

натиснути на кнопку "Розбити"

У вікні, що відображає вибраний рядок і результат його розбиття, натиснути на кнопку "Прийняти".

Вибір колонок, що переглядаються здійснюється шляхом натискання на назви колонок у списку, що випадає.

# 2 ЕКОНОМІЧНИЙ РОЗРАХУНОК

## 2.1 Резюме

Темою даного дипломного проекту є «Розробка десктопного програмного забезпечення для системи розподілу педагогічного навантаження за рахунок коштів платформи .NET». Було розроблено інформаційне та програмне забезпечення комплексу завдань, які дозволяють скоротити трудомісткість виконуваних робіт та вартісні витрати на їх виконання.

Ефективність кожного програмного продукту визначається його якістю та ефективністю процесу розробки. Якість ПП визначається наступними складовими: з точки зору користувача; з позиції використання ресурсів; виконання вимог до програмного забезпечення.

Оцінка якості програмного продукту з точки зору користувача надала такі переваги порівняно із звичайним документообігом:

1. Можливість оперативного контролю за достовірністю інформації;
2. Зменшення кількості можливих помилок при генеруванні похідних даних;
3. Можливість швидкого доступу до будь-яких даних;
4. Економія пам'яті в процесі складання запиту для зв'язування та управління даними звіту.

Оцінка якості програмного продукту включає визначення трудомісткості і вартості його створення.

**2.2. Визначення трудомісткості розробки програмного забезпечення.**

Тривалість розробки програмного продукту залежить від його обсягу, трудомісткості розробки, кваліфікації виконавців, а також планових термінів, визначених умовами ринку. Методом структурної аналогії по відповідних каталогах аналогів програмного забезпечення визначаємо обсяг програмних засобів, у тисячах умовних машинних команд програми аналога

У таблиці 4.1 представлені аналоги програмного забезпечення, функції яких, у більшому або меншому ступені, виконує розроблений програмний продукт.

Каталог аналогів

Таблиця 4.1

| Найменування ПП | Обсяг функції ПП – $V_o$, усл. машинних командах. |
|---|---|
| 1. ПП СУБД | 2460 – 9360 |
| 2. Комплексні системи ведення БД | 930 – 7320 |
| 3. ПП організації обчислювального процесу | 13000 – 10400 |

Для нашого варіанта виділено сірим кольором.

Вибравши аналог ПП, що містить Vo в умовних машинних командах, трудомісткості визначати на основі табл.4.2

Таблиця.4.2

| Обсяг ПП, тис.умов.машинних команд | Норма часу, люд/год |
|---|---|
| 1.00 | 215 |
| 2.00 | 233 |
| 3.00 | 258 |

На підставі отриманого значення, по довіднику, визначається укрупнена норма часу на розробку аналога програмного забезпечення (коректується поправочним коефіцієнтом враховуючої умови розробки ПП, тобто в умовах комп'ютера, Кк=0,7÷0,8): $T^a = 215 \times 0,8 = 172$ (люд/годин).

Трудомісткість програмного продукту визначається по кожному етапу розробки окремо на підставі трудомісткості аналога з урахуванням складності розробки, ступеня новизни і ступеня використання в розробці стандартних модулів на підставі формул:

$$T_{ТЗ} = T^a p \times L_1 \times K_H \qquad (4.1)$$

$$T_{ТП} = T^a p \times L_2 \times K_H \qquad (4.2)$$

$$T_{РП} = T^a p \times L_3 \times K_H \times K_T \qquad (4.3)$$

Для розрахунку необхідні наступні коефіцієнти:

Li – питома вага і-го етапу розробки (див. табл. 4.2.);

Кн – поправочний коефіцієнт, що враховує ступінь новизни (див. табл. 4.3.);Кт – поправочний коефіцієнт, що враховує ступінь використання в розробці типових програм (див. табл. 4.4.).

Таблиця 4.2.Значення питомих коефіцієнтів трудомісткості стадії в загальній трудомісткості розробки ПП.

| Код стадії | Ступінь новизни | | |
|---|---|---|---|
| | А | Б | В |
| ТЗ ($L_1$) | 0,15 | 0,12 | 0,12 |
| ТП ($L_2$) | 0,16 | 0,15 | 0,11 |
| РП ($L_3$) | 0,55 | 0,58 | 0,61 |

Для нашого варіанта виділено сірим кольором.

Таблиця 4.3. Значення поправочного коефіцієнта, що враховує ступінь новизни

| Кодступеня новизни | Ступінь новизни | Значення $К_н$ |
|---|---|---|
| А | Принципово нові ПО | 1,75 – 1,2 |
| Б | ПО – розвиток визначеного параметричного ряду | 1,0 – 0,8 |
| В | ПО маючий аналог | 0,7 |

Для нашого варіанта виділено сірим кольором.

Таблиця 4.4. Значення коефіцієнта ступеня використання в розробці типових програм

| Ступінь охоплення реалізованих функцій розроблювального ПО типовими програмами, % | Значення $К_т$ |
|---|---|
| 60 і вище | 0,6 |
| 40-60 | 0,7 |
| 20-40 | 0,8 |
| До 20 | 0,9 |

Для нашого варіанта виділено сірим кольором.

Тепер розраховуємо трудомісткість по кожному етапу окремо:

Трудомісткість технічного завдання

$$Т_{тз}=Т^а *L_1*К_н= 172 *0,12*0,7 = 14,44 \text{ (люд/годин)} \quad (4.1)$$

Трудомісткість розробки технічного проекту

$$Т_{тп}=Т^а *L_2*К_н =172*0,11*0,7 = 13,24 \text{ (люд/годин)} \quad (4.2)$$

Трудомісткість розробки робочого проекту

$$Т_{рп}=Т^а *L_3*К_н*К_т =172*0,61*0,7*0,7=51,41 \text{ (люд/годин)} \quad (4.3)$$

Для подальших розрахунків визначили кількість папера, витраченого на кожен етап: технічне завдання $N_{тз}$= 2 (стор), розробка ТП $N_{тп}$=18 (стор),

розробка робочого проекту $N_{рп}$=25(стор), пояснювальна записка відповідно $N_{пз}$= 10 (стор) Розрахунок зведений у таблицю 4.5

Таблиця 4.5. Розрахунок трудомісткості ПП

| Найменування етапів | Розрахунок, годин. | | |
|---|---|---|---|
| 1 | 2 | 3 | 4 |
| 1.ТЗ | $T_{Pтз}$=14,44 | $T_{кк}=0,7*N_{тз}$= 0,7*2=1,4 | $T_{нк}=0,15*N_{тз}$=0,15*2=0,30 |
| 2.Розробка ТП | $T_{Pтп}$=13,24 | $T_{кк}=0,7*N_{тп}$=0,7*18=12,6 | $T_{нк}=0,15*N_{тп}$=0,15*18=2,7 |
| 3.Розробка РП | $T_{Pрп}$=51,41 | $T_{кк}=0,7*N_{рп}$=0,7*25=17,5 | $T_{нк}=0,15*N_{рп}$=0,15*25=3.8 |
| 4.Розробка ПЗ | $T_{пз}=1,5*$ $*N_{пз}=1,5*10$ =15 | $T_{кк}=0,7*N_{тз}$=0,7*10=7 | $T_{нк}=0,15*N_{пз}$=0,15*10 =1,5 |
| Усього, в т.ч.: | ΣT=140,9 | | |
| - на розробку | $ΣT_р$=94,1 | | |
| - контроль керівника | | $ΣT_{кк}$=38,5 | |
| - нормоконтроль | | | $ΣT_{нк}$=8,3 |

**2.3 Розрахунок ціни програмного продукту.**

У цьому розділі для визначення ціни розраховуємо основну заробітну плату виконавців, матеріальні витрати, вартість машино – години і витрати на розробку ПО. Розрахунок основної заробітної плати виконавців приведений у таблиці 4.6. Відповідно до статті 8 «Закону про Державний бюджет України на 2022» встановлено мінімальну заробітну плату у місячному розмірі з 1 січня 2022 року - 6500 гривень; мінімальну погодинну тарифну ставку – 39.26 грн.

Таблиця 4.6 Розрахунок основної заробітної плати виконавців.

| Найменування робіт | Трудомісткість робіт, години | Погодинна тарифна ставка, грн. | Розрахунок, грн. |
|---|---|---|---|
| 1.Розробка ПП | 94,1 | 39.26 | 3 694,36 |
| 2.Контроль керівника | 38,5 | 60.00 | 2310 |
| 3.Нормоконтроль | 8,3 | 60.00 | 498 |

| | | | |
|---|---|---|---|
| Усього | - | - | ΣЗо = 6502,36 |

Зробимо розрахунок матеріальних витрат на розробку ПП. Розрахунок зведемо в таблицю 4.7

Таблиця 4.7 Розрахунок матеріальних витрат на розробку ПО

| Найменування матеріальних витрат | Тип, модель | Кількість | Ціна одиниці, грн. | Вартість, грн. |
|---|---|---|---|---|
| Папір | Лист А4 | 50 | 2,5 | 125 |
| Разом | - | - | - | $В_{м\,і}$ = |
| Транспортно– заготівельні витрати (10%) | | | | $Втр\_з = 0{,}1 \times Вм1$ = 12,5 |
| Усього | | | | $В_м = В_{мі} + В_{тр\_з}$ = 137,5 |

На підставі отриманих даних по окремих статтях витрат складена калькуляція планової собівартості в цілому ПП за формою, приведеною в таблиці 4.8.

Таблиця 4.8. Розрахунок статей витрат планової собівартості

| Стаття витрат | Значення, грн. | Формула розрахунку |
|---|---|---|
| 1. Матеріали | 137,5 | $В_м$ (див. табл. 4.7) |
| 2. Основна заробітна плата | 6502,36 | $З_о$ (див. табл. 4.6) |
| 3.Додаткова заробітна плата | 975,35 | $Зд = 0{,}15 \times Зо$ =0,15*6502,36 |
| 4.Відрахування до єдиного фонду соціального внеску | 1645,09 | $Вє.с.в. = 0{,}22 \times (Зо + Зд)$ =0,22* (6502,36+975,35) |
| 5. Накладні витрати | 1950,70 | $Внак. = 0{,}3 \times Зо$ =0,3*6502,36 |
| 6. Повна собівартість | 11211 | $С_{пов} = В_м + З_о + З_д$ +Вє.с.в. + Внак.= |

Розмір прибутку, що включається в ціну, визначаємо по наступній формулі:

$П=(С_п*Р)/100 = (11211*10)/100 = 1121{,}1$ грн (4.4)

Де р – плановий рівень рентабельності (10-15%).

Оптова ціна (кошторисна вартість) визначається по формулі:

$Ц_о = С_{пов} + П =$ 11211 + 1121,1 грн = 12332,10 грн (4.5)

Податок на додану вартість визначаємо по наступній формулі:

ПДВ = 0.2 * $Ц_о$ = 0,2*12332,10 = 2466,42 грн (4.6)

Виходячи з отриманих даних, ціна реалізації розробленого програмного продукту на основі наступної формули, становитиме:

$Ц_р = Ц_о + ПДВ = 12332,10 + 2466,42 = 14798,52$ грн (4.7)

# 3 ОХОРОНА ПРАЦІ

**Вступ**

Умови та безпека праці, їх стан та покращення – самостійна і важлива задача соціальної політики будь-якої сучасної промислово розвинутої держави, яку вирішує така невід'ємна складова БЖД, як охорона праці. Рівень безпеки будь-яких робіт у суспільному виробництві значною мірою залежить від рівня правового забезпечення цих питань, тобто від якості та повноти викладення відповідних вимог в законах та інших нормативно-правових актах. Для вирішення існуючих проблем в сфері охорони праці необхідна ефективна взаємодія всіх органів державної влади та громадськості, а також реалізація як на державному, так і на місцевих рівнях відповідних програм, спрямованих на корінне покращення умов і охорони праці.

В даному розділі розглянуті питання безпечних та здорових умов праці програміста при розробці десктопного програмного забезпечення для системи розподілу педагогічного навантаження.

## 3.1 Аналіз небезпечних та шкідливих чинників, що впливають на працівника.

Небезпечні і шкідливі виробничі чинники здатні зробити згубний вплив на здоров'ї офісного працівника, і його працездатність.

Аналіз і оцінка стану умов та безпеки праці - це обов'язкова складова роботи керівництва підприємства щодо планування відповідних заходів з охорони праці. Тривала дія на людину шкідливого виробничого чинника призводить до захворювання.

До факторів ризику при роботі з ПК відносяться:

- проблеми, пов'язані з електромагнітним випромінюванням;
- проблеми зору;
- проблеми, пов'язані з м'язами і суглобами; проблеми безсоння, стресів, нервових розладів;
- проблеми органів дихання.

## 3.2 Розробка заходів з охорони праці

### 3.2.1 Виробничі приміщення

Приміщення для розміщення робочих місць з ВДТ ЕОМ і ПЕОМ не можуть знаходитись у підвальних приміщеннях та на цокольних поверхах. Площа на одне робоче місце становить не менше ніж 6,0 м$^2$, а об'єм – не менше ніж 20,0м$^3$.

Виробниче середовище характеризується такими параметрами як параметри мікроклімату, освітлення, шуму і вібрації, рівнів електромагнітного та іонізуючого випромінювання.

### 3.2.2 Мікроклімат робочої зони працівників, вентиляція.

У виробничих приміщеннях на робочих місцях мають забезпечуватись оптимальні значення параметрів мікроклімату, враховуючи характер роботи працівника – сидячи, без фізичного напруження. Ці значення параметрів повинні становити: температура повітря, не більше 22-24$^0$С; відносна вологість 40-60%; швидкість руху повітря 0,1 м/с. Для досягнення вказаних параметрів у приміщенні встановлені системи опалення та вентиляції. Вентиляція – припливно-витяжна та кондиціонер. Встановлені вимоги до параметрів мікроклімату виконанні.

### 3.2.3 Освітлення робочого місця, шум, вібрація

Відповідно до Правил в приміщенні з ПК використовується природне та штучне освітлення. Штучне освітлення здійснюється системою загального рівномірного освітлення. Значення освітленості на поверхні робочого столу в зоні розміщення документів становить 300-500 лк. Як джерела світла для штучного освітлення застосовані люмінесцентні лампи типу ЛБ.

### 3.2.4 Електробезпека.

Використання електричної енергії на виробництві повезене з небезпекою дії електроструму на організм людини.

Ураження струмом може виникнути при роботі під напругою і при несправному стані електроустановок, а саме при дотику до оголених проводів, незаземлених металевих корпусах електричного обладнання, при відкритих рубильниках і других струмоведучих частинах.

Відповідно до ГОСТ 12.1.019-79 «Электробезопасность. Общие требования» електробезпечність людини повинна забезпечуватися конструкцією електроустановок, технічними засобами і засобами захисту, організаційними і технічними заходами. Для захисту працюючих від ураження електричним струмом передбачені наступні заходи:

- недоступність струмоведучих частин;
- захисне заземлення (занулення) корпусів електрообладнання;
- передбачені рубильники закритого типу;
- блокіровка, надписи, плакати, засоби індивідуального захисту (калоші і боти діелектричні (ГОСТ 13385-78), рукавиці резинові діелектричні, коврики резинові діелектричні (ГОСТ 4997-75).

Заземлені конструкції, що знаходяться в приміщеннях, де розміщені робочі місця операторів ( батареї опалення, водопровідні труби, кабелі із заземленим відкритим екраном) мають бути надійно захищені діелектричними щитками або сітками з метою недопущення потрапляння працівника під напругу.

У приміщенні, де одночасно експлуатуються понад п’ять ЕОМ, на помітному та доступному місці встановлюється аварійний резервний вимикач, який може повністю вимкнути електричне живлення приміщення, крім освітлення.

ЕОМ з ВДТ і ПК повинні підключатися до електромережі тільки за допомогою справних штепсельних з’єднань і електророзеток заводського виготовлення. У штепсельних з’єднаннях та електророзетках, крім контактів фазового та нульового робочого провідників, мають бути спеціальні контакти для підключення нульового захисного провідника. Їхня конструкція має бути такою, щоб приєднання нульового захисного провідника відбувалося раніше, ніж приєднання фазового та нульового робочого провідників. Порядок роз’єднання при відключенні має бути зворотнім.

Не допускається підключати ЕОМ з ВДТ і ПК до звичайної двопровідної електромережі, в тому числі – з використанням перехідних пристроїв.

Штепсельні з’єднання та електророзетки для напруги 12В та 42В за своєю конструкцією та візуально ( за кольором) мають відрізнятися від штепсельних з’єднань для напруги 127В та 220В.

При організації робочих місць операторів електромережу штепсельних розеток для живлення ЕОМ з ВДТ і ПК у центрі приміщення прокладають у каналах або під знімною підлогою в металевих трубах або гнучких металевих рукавах. При цьому не допускається застосовувати провід і кабель в ізоляції з вулканізованої гуми та інші матеріали, які містять сірку.

ЕОМ з ВДТ та ПК закордонного виробництва повинні відповідати вимогам національних стандартів держав-виробників і мати відповідну позначку на корпусі, в паспорті або в іншій експлуатаційній документації.

### 3.2.4 Організація робочого місця користувача ПК

Обладнання і організація робочого місця з ВДТ забезпечують відповідність конструкції всіх елементів робочого місця та їх взаємного розташування ергономічним вимогам з урахуванням характеру і особливостей трудової діяльності ( ГОСТ 12.2.032-78, ГОСТ 22.269-76, ГОСТ 21.889-76).

Конструкція робочого місця користувача ВДТ має забезпечити підтримання оптимальної робочої пози. Робочі місця слід розташовувати відносно світлових прорізів, щоб природне освітлення падало збоку, переважно зліва.

При розміщуванні робочих столів з ВДТ слід дотримуватись таких відстаней: між бічними поверхнями ВДТ – 1,2м; від тильної поверхні одного ВДТ до екрана іншого – 2,5м.

Екран ВДТ має розташовуватися на оптимальній відстані від очей користувача, що становить 600…700 мм, але не ближче ніж за 600 мм з урахуванням розміру літерно-цифрових знаків і символів.

Розташування екрана ВДТ має забезпечувати зручність зорового спостереження у вертикальній площині під кутом $+30^0$ до нормальної лінії погляду працюючого.

Клавіатуру слід розташовувати на поверхні столу на відстані 100…300 мм від краю, звернутого до працюючого. У конструкції клавіатури має передбачатися опорний пристрій, який дає змогу змінювати кут нахилу поверхні клавіатури в межах $5…15^0$.

Умови організації робочого місця користувача ВДТ в дипломній роботі виконані.

## 3.3 Пожежна безпека.

Основними причинами пожежі є: необережне поводження з вогнем, незадовільний стан електротехнічних установок і невиконання правил їх експлуатації, несправність виробничого обладнання і порушення режимів технологічних процесів, порушення правил пожежної безпеки.

До засобів гасіння пожежі відносяться внутрішні пожежні водопроводи (крани - ПК), вогнегасники (вуглекислотні та порошкові), сухий пісок тощо.

В будівлях пожежні крани встановлюють в коридорах, на майданчиках сходових кліток. Кожний пожежний кран укомплектований пожежним рукавом і розміщений у відповідних ящиках, які знаходяться на висоті 1,35 м від полу.



Для гасіння пожеж на початкових стадіях широко застосовуються вогнегасники. У виробничих приміщеннях це головним чином вуглекислотні вогнегасники, достоїнством яких є висока ефективність гасіння пожежі, збереження електричного устаткування. Розташовують вогнегасники на видних місцях, на висоті не більше як 1,5 м від полу. Виробничі приміщення мають запасні виходи. Двері повинні мати освітлений надпис «Запасний

вихід». План евакуації вивішується на видному місці у основного виходу із приміщення.

# ВИСНОВКИ

У процесі виконання дипломної роботи мети цього проекту було досягнуто. В результаті виконання проекту було проведено такі роботи:

Вибрані необхідні засоби розробки;

- Розроблено архітектуру взаємодії з модулем "Педагогічна навантаження"
- Розроблено архітектуру пакетів системи;
- Створено необхідні таблиці у базі даних для модуля "Педагогічна навантаження";
- Виявлено проблеми непрацюючого та незатребуваного функціоналу та враховано у розробленому модулі "Педагогічна навантаження"

У перспективі планується впровадити модуль "Учбове навантаження" в Одеському технічному професійному коледжі.

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. Web Call Server 5: [Электронный ресурс]. URL: https://flashphoner.com/

2. 40 open source, free and top UML tools: [Электронный ресурс]. URL: https://www.predictiveanalyticstoday.com/open-source-free-unified-modeling-language-uml-tools/

3. Что такое ASP.NET: [Электронный ресурс]. URL: http://www.internet-technologies.ru/articles/lekciya-1-chto-takoe-asp-net-installyaciya-i-testovyy-proekt.html

4. Web Call Server - Руководство Разработчика: [Электронный ресурс]. URL: https://flashphoner.com/docs/wcs5/wcs_docs/html/ru/wcs-developer-guide-2/

5. Which is Best for Web Application Development—Dot Net, PHP, Python, Ruby, or Java: [Электронный ресурс]. URL: https://www.addonsolutions.com/blog/which-is-best-for-web-application-development-dot-net-php-python-ruby-or-java.html/ (Дата обращения 15.04.2018);

6. Сравнение современных СУБД: [Электронный ресурс]. URL: http://drach.pro/blog/hi-tech/item/145-db-comparison/

7. Версии лицензий в линейке продуктов Oracle Database : [Электронный ресурс]. URL: https://oracle-patches.com

8. Выпуски и компоненты SQL Server 2014 [Электронный ресурс]. URL: https://msdn.microsoft.com/ru-ru/library/ms144275%28v=sql.120%29 (Дата обращения 17.04.2018)