Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра програмної інженерії
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

магістр
(назва освітнього ступеня)

на тему: «Розробка універсальної мобільної Web-системи автоматизації обліку ресурсів бізнес-структури»

Виконала: студентка VI курсу, групи СП-61
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва спеціальності)

(підпис) Деркач В.В. (прізвище та ініціали)

Керівник (підпис) Цуприк Г.Б. (прізвище та ініціали)

Нормоконтроль (підпис) Стоянов Ю.М. (прізвище та ініціали)

Завідувач кафедри (підпис) Петрик М.Р. (прізвище та ініціали)

Рецензент (підпис) (прізвище та ініціали)

Тернопіль 2024

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра програмної інженерії
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Петрик М.Р.
(підпис) (прізвище та ініціали)
«_____» _____ 2024 р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня магістр
(назва освітнього ступеня)

за спеціальністю 121 «Інженерія програмного забезпечення»
(шифр і назва спеціальності)

студентці Деркач Вікторії Василівні
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка універсальної мобільної Web-системи автоматизації обліку ресурсів бізнес-структури

Керівник роботи Цуприк Галина Богданівн, к.т.н., доц.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «16» грудня 2024 року № 4/7-1174
2. Термін подання студентом завершеної роботи 10 грудня 2024 р.
3. Вихідні дані до роботи Завдання на розробку системи

4. Зміст роботи (перелік питань, які потрібно розробити)
Сформулювати, погодити та затвердити тему кваліфікаційної роботи;
розробити та затвердити завдання;
проаналізувати завдання, зробити огляд та проаналізувати бібліографічні матеріали щодо стану справ та тенденій в обраному напрямку дослідження;
сформулювати завдання, обґрунтувати цілі дослідження;
розробити та спроєктувати систему;
описати технології, етапи розробки та саму розробленупрограмну систему (у разі потреби)
розробити план тестування програмного продукту; оформити допоміжну документацію;
проаналізувати роботу щодо питань з дотримання положень про охорону праці та безпеку в надзвичайних ситуаціях; оформити пояснювальну записку;
зробити відповідні висновки за результатами виконаної роботи.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
Ілюстративна частина кваліфікаційної роботи оформляється у вигляді слайдів, висвітлює основні розділи та етапи роботи та містить:
тему роботи, мету, предмет та об'єкт дослідження,
сформульовані завдання, перелічено етапи виконання кваліфікаційної роботи, короткий аналіз актуальності теми, варіант архітектури програмної системи,
представлено опис програмного забезпечення; опис програмного забезпечення та тестування.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	Осухівська Г.М.		
Безпека в надзв. ситуаціях			
Нормоконтроль	Стоянов Ю.М.		

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Вибір та погодження теми з керівником. Створення та затвердження технічного завдання	17.06.2024-23.06.24	виконано
2.	Аналіз технічного завдання, підбір та аналіз	24.06-	
	бібліографічних матеріалів необхідних для виконання	01.07.2024	виконано
	кваліфікаційної роботи		
3.	Проєктування та розробка програмного забезпечення	21.09.2024	виконано
4.	Тестування та дослідна експлуатація програмного		
	забезпечення	16.11.2024	виконано
5.	Створення допоміжної документації. Написання та		
	перевірка на відповідність вимогам пояснювальної		
	записки	07.12.2024	виконано
6.	Апробація результатів роботи	19.12.2024	виконано
7.	Перевірка програмою антиплагіат	08.12.2024	виконано
8.	Охорона праці	.12.2024	виконано
9.	Безпека в надзвичайних ситуаціях	.12.2024	виконано
10.	Формування записки кваліфікаційної роботи	.12.2024	виконано
11.	Нормоконтроль	.12.2024	виконано
12.	Попередній захист	10.12.2024	виконано
13.	Рецензування кваліфікаційної роботи	.12.2024	виконано
14.	Захист кваліфікаційної роботи	26.12.2024	

Студентка _____ Деркач В.В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Цуприк Г.Б.
(підпис) (прізвище та ініціали)

РЕФЕРАТ / ABSTRACT

Атестаційна робота ОР «магістр» містить:

_____с., _____рис., _____табл, _____джер.

КВАЛІФІКАЦІЙНА РОБОТА, ПРОГРАМНИЙ ПРОДУКТ, ТЕХНОЛОГІЇ РОЗРОБКИ, ПРОЄКТУВАННЯ, КОНСТРУЮВАННЯ, ПЛАН ТЕСТУВАННЯ, ОХОРОНА ПРАЦІ

Розроблене програмне забезпечення, призначене для автоматизації робочого часу працівників підприємства, на час захисту дана робота перебуває на етапі впровадження в експлуатацію. Тестові випробування системи показали її спроможність вирішувати бізнес-задачі, що були початково заявлені у вимогах замовника. Крім того, завдяки ООП забезпечено можливості модифікації та розширення системи у випадку зміни вимог замовника.

В роботі представлений аналіз предметної області, обґрунтовано вибір технології розробки, представлено основні процеси проєктування та конструювання системи, розроблено план та проведено тестування.

QUALIFICATION WORK, SOFTWARE PRODUCT, DEVELOPMENT TECHNOLOGIES, DESIGN, CONSTRUCTION, TESTING PLAN, LABOR SAFETY

The developed software is intended for the automation of the working time of the enterprise's employees, at the time of defense this work is at the stage of implementation into operation. Test tests of the system showed its ability to solve business problems that were initially stated in the customer's requirements. In addition, thanks to the OOA, the possibility of modifying and expanding the system in the event of a change in the customer's requirements is provided.

The work presents an analysis of the subject area, justifies the choice of development technology, presents the main processes of system design and construction, develops a plan and conducts testing.

ЗМІСТ

ВСТУП

Автоматизована інформаційна система (системи) є одним із невід'ємних атрибутів бізнес-структур всіх цивілізованих країн, в тому числі і країн Західної Європи вже впродовж. без перебільшення, кількох останніх десятиліть. Саме їх використання підвищує ефективність, а заразом із цим і прибутковість бізнесу, тим самим збільшуючи і його конкурентоспроможність. Якщо ж розглядати світовий ринок розробок програмного забезпечення в цілому то, на сьогоднішній день, то з відкритих джерел можна контактувати, що успішно працюють цілий ряд компаній, які спеціалізуються на розробці програмних систем рівня корпорації. Для розробки таких систем розвивається спеціальний специфічний цілий комплекс технологій, який і отримав всім вже відому назву «об'єктно-орієнтовний аналіз та проєктування». Не дивлячись на складний , в плані ведення будь-якої успішно діяльності час, я вірю, що вже дуже скоро війна завершиться і буду опиратись на той факт, що сучасна динаміка розвитку бізнес-структур в саме нашій країні вимагатиме, і в найкоротший час, впровадження інформаційних систем для автоматизації їх діяльності. Саме на цьому я і ґрунтувалась при виборі моєї теми та вважаю її актуальною актуальною адже пов'язана вона саме із розробкою програмних систем подібного класу та призначення.

Важливою характеристикою сучасних автоматизованих корпоративних систем призначених для застосування саме бізнес-структурами можна назвати їхню мобільність та загальну доступність. Для забезпечення цих таких вимог досить добре, на мою думку, підходить середовище типу Web, оскільки, зазвичай, системи такого класу володіють Web-інтерфейсом, що робить їх мобільними та дає змогу використовувати в буд-якій точці світу де є доступ до світової мережі.

Враховуючи потреби військового часу та поствоєнного національного ринку такого продукту як програмне забезпечення та визначаючи Web як

всеохоплююче інформаційне середовище, я і прийшла до формулювання обраної мною теми кваліфікаційної роботи, оскільки актуальною є розробка програмного забезпечення саме із підтримкою Web-інтерфейсу і конкретно для вдосконалення і пришвидшення, без будь-яких втрат, через автоматизацію, діяльності бізнес-структур.

Щодо завдання, яке повстає перед управлінським персоналом бізнесової структури при чому не лише великого, але й середнього і малого бізнесу, природною ціллю якого є розвиток та конкурентоспроможність, то тут все просто: воно повинно фокусуватись на розробку та використання ефективної інформаційної системи обліку саме робочого часу. А вже використання інформаційних систем подібного типу ґрунтується на знаннях організації, управління та інформаційної технологій, що формують систему. В умовах сьогодення досить швидко зростають вимоги щодо якості інформаційних систем, однією із основних функцій яких є надання достовірної й що важливо максимально повної інформації як працівникам, так і всім, хто пов’язаний із управлінням діяльністю підприємства.

Таким чином метою та завдання представленої далі роботи є:

1 – проведення аналізу бізнесових процесів, пов’язаних із обліком часового ресурсу працівників всіх структур підприємства;

2 – окреслення призначення та функціональних можливостей розроблюваної системи;

3 – розробка програмного забезпечення, яке представлятиме собою цілісну Web-систему, призначену для робочого часового ресурсу;

4 – розробити та надати інструкції по використанню реалізованої програми.

Об’єктом дослідження я вбачаю автоматизацію за допомогою Web-програм.

Предметом дослідження є безпосередня розробка універсальної мобільної Web-системи автоматизації обліку ресурсів бізнес-структури.

1 РОЗРОБКА ПРОГРАМНОЇ СИСТЕМИ

1.1 Аналіз вимог до програмної системи

1.1.1 Аналізування предметної області

Основним завданням яке я ставлю до розроблюваної мною системи є автоматизація обліку ресурсів на прикладі одного із непоновлюваних основних – обліку тривалості роботи та заробітної оплати відповідно працівників бізнес-структури. В цілому ж варта зазначити, що робота з даними є важливою задачею, і проблеми з цим стосуються якщо не всіх, то більшості напрямків діяльності [1-5].

Розроблюваний програмний продукт представлятиме собою Web-систему, що призначена для використанням адміністраторами та працівниками бізнес-структури, наприклад, підприємства. Зокрема, адміністратори мають право для встановлення часу по кожному працівникові, кожен із яких, відповідно, повинен звітуватися за виконану упродовж дня роботу відповідно до запланованої.

До основних завдань обліку як часу затраченого на певний об’єм робіт та відповідної заробітної плани є:

1 – коректне фактичному встановлення виробітку та затраченого на виконання видів робіт часу;

2 – отримання відомостей про виконання норм кожним працівником;

3 – точне, коректне та своєчасне нарахування заробітку для кожного із працівників;

4 – систематична перевірка та контроль коректності та відповідності використання фонда заробітної плати.

До основних передумов автоматизації цього напрямку для обліку є: безпека [6,7] суттєвий об’єм та види форм як вхідної так і вихідної документації; висока трудомісткість такого збору, реєстрації так само як і опрацювання інформації, багатогранність задач, що потребують розв’язку та їх

значимість, значні об’єми вхідної інформації необхідної для коректного нарахування заробітної плати, стягнень та утримань; обчислювальна складність виконання таких розрахунків.

В галузі інформаційних технологій використовується підхід типу погодинної форми оплати за певний встановлений об’єм визначених видів робіт, який здійснюється за відпрацьований час відповідно до тарифної ставки ІТ-працівника. Такий підхід також є специфічним для розрахунків, оскільки за один і той самий час протягом дня працівник може отримувати різну оплатню, відповідно до об’єму та повноти виконаної роботи. Така система оплати визначається на основі відповідної документації, яка відрізняється від тої, що використовується за відпрацьований робочий день в кількості годин визначеній законодавством.

Таким чином основна задача, яку я ставлю перед собою першочергово полягає в правильно організованій структурі зберігання інформації (тобто структурі баз даних), алгоритмів внесення (доповнення), зчитування, корегування інформації. Сама ж програма, при цьому, являтиме собою різновид довідникової системи .

Таким чином для цього етапу мені було необхідно визначитись та сформулювати вимоги до розроблювальної мною системи, зокрема: -

1 – чіткість та логічна структура баз даних;

2 – наявність, як мінімум, третьої нормальної форми для усіх створюваних структур даних;

3 – присутність логічно послідовних зв’язків між компонентами створюваної структури даних;

4 – способи тримання інформації з системи.

На сьогодні існує велика кількість інформаційних систем автоматизації діяльності бізнес-структур, проте, багато із них використовують дещо специфічні методи роботи та не завжди мають змогу використовувати існуючі програми такого типу. Моєю задачею я вважаю спробу розробити такий продукт, який би задовільняв вимоги більшості бізнес-структур, які працюють

прозоро та в рамках правового поля і правильний прийом і оформлення первинної документації при цьому є надійною основою для забезпечення високоефективної діяльності структури [8-13].

1.1.2 Визначення та постановка задачі

Після того як я провела аналіз предметної області та виявлення основних проблем потрібно визначитись та окреслити завдання, які потрібно вирішити першочергового [1]. Першим і, на мою думку, найголовнішим завдання є те, якому типу бази даних надати перевагу. Яким способом буде здійснюватись зв’язок із базою даних. Чи потрібні якісь додаткові данні для внесення їх у базу, чи достатньо попередньо вже перелічених. Які ще процеси можна автоматизувати в обраній предметній області. Чи взагалі є потреба в розробці заявленої інформаційної системи в принципі. Які процеси доцільно і вони реально потребують покращення. Доцільність додавання додаткових функцій, та як до них поставляться зацікавлені користувачі. Які додані функції будуть корисні кінцевому користувачеві, і доцільність їх додавання, чи не будуть вони дублювати існуючі. Яким буде інтерфейс, і чи зручним він буде. Чи достатньо буде сутностей, які виявлено в процесі аналіза предметної області, чи, можливо, є зайві, чи, навпаки, їх занадто мало. Чи в повній мірі і достатньо вивчено предметну область, чи необхідно провести додатковий аналіз. Насправді цей перелік я можу продовжувати ще дуже довго, але час відведений на виконання кваліфікаційної роботи обмежено і потрібно виявити основні проблеми, все інше залишити на доопрацювання і можливість покращувати систему в майбутньому.

Таким чином, в результаті детального аналізу я виявила та сформулювала перелік завдань, до яких віднесла:

- тип бази данних;

- як відбуватиметься з’єднання із базою даних;

- які саме дані будуть зберігатися;

- впровадження процесів необхідних для якісної автоматизації та покращення;

- необхідні функції на стороні користувача;

- сутності запропонованої бази даних;

- достатня оптимальна кількість інформації.

Далі пропоную визначитись з даними котрі мені будуть потрібні для роботи бази даних, основною інформацією, до якої я віднесу:

- інформація про працівників;

- стосовно підрозділів підприємства;

- стосовно посад працівників;

- стосовно тарифних окладів для кожного із працівників;

- стосовно обліку затрат часу кожного окремого працівника.

Таким чином розроблюваний програмний продукт має задовольняти наступні вимоги, зокрема:

- можливість щоденної звітності працівників за виконану роботу;

- можливість облікувати затрати часу кожного окремого працівника;

- можливість автоматизованого розрахунка заробітної плати;

- можливість додавати, міняти, видаляти дані про працівників, їх, посади, підрозділи;

- можливість та інструменти для фільтрування та сортування даних;

- можливість забезпечити стабільно надійне зберігання даних;

- наявність зручного користувацького інтерфейсу.

Таким чином виокреслюється дещо змінена мета розробки запропонованої системи, а саме є прискорення розрахунків заробітної плати працівників через автоматизацію.

Основні запити, які формуються клієнтом будуть здійснюватися інструментами графічного інтерфейсу. До основних запитів належать:

- вибірка та сортування даних, для полегшення сприйняття користувачем;

- запити на внесення і зміну данних, які також будуть виконуватися системою, після необхідності заповнити відповідні поля для внесення. Відповідні данні будуть надсилатися на сервер.

Оскільки передбачається, що кінцевим користувачем буде використовуватись лише графічний інтерфейс, то виникає потреба в забезпеченні його інтуїтивності, зробити його максимально пристосованим до потреб кінцевого користувача з різним рівнем та досвідом володіння комп'ютером та роботою web – ресурсів.

З цим баченням далі я перелічу вимоги до інтерфейсу, зокрема:

- сумісність поєднану із потребами і можливостями кінцевого користувача;

- забезпечення простоти при переході від виконання однієї із функцій до виконання іншої;

- допустимі запити від користувача повинні мати чітке і однозначне формулювання;

- забезпечення однозначної, зрозумілої та, по можливості, простої реакції системи на всі види та типи запитів;

- максимальна простота при його використанні;

- можливість адаптації до розв'язування конкретно поставленої задачі;

- оптимальна кількість деталей, не перевантаженість; не повинно бути нічого зайвого, окрім необхідного для розуміння результату;

- відсутність зайвих відволікаючих деталей, які не несуть інформаційного змісту, а слугують лише як прикраса та відволікатимуть від основної задачі;

- надання користувачеві чітких та однозначних вказівок стосовно можливих наступних дій;

- можливість зворотний зв'язку із користувачем та відповіді на його запити;

- можливість використання користувачем різних, взаємодоповнюючих формам представлення результатів, залежно від типів запитів та/ чи від характеру отриманих рішень;

- консистентність інтерфейсу, тобто, йдеться про те, що візуалізація ґрунтуватиметься на використанні відомих та загально використовуваних інструментів призначених для представлення інформації.

1.1.3 Пошук актантів та варіантів використання [14,15,16]

З попередньо окреслених вимог я сформулюю завдання, які планую виконати. Зокрема:

1- необхідно якимось чином надавати загальну інформацію про табель обліку використаного робочого часу і тарифного окладу та відповідної йому заробітньої плати для кожного із працівників.

2- Потрібно, в одному місці, зібрати інформацію для можливості ефективно нею користуватись.

3- Необхідно забезпечити ефективні доступ, збереження, редагування інформації. Метою цих заходів є зменшення невизначеності, неточності інформації.

4- Варто передбачити можливість додавання і редагування а заразом і видалення тарифного окладу і відповідної йому заробітної оплати та табелю обліку використаного робочого часу. Тут метою є забезпечити зв’язок між адміністрацією та працівниками.

5- Потрібно визначити, які саме переваги та доступи отримає кожен із користувачів, але перед тим необхідно встановити, хто є тими типовими користувачами. Логічно, що це були б адміністратори та працівники.

Адміністратор – той хто вводить інформацію про працівників, їх посади, організаційні підрозділи, редагує, переносить або ж видаляє. До його функцій також віднесу введення тарифної ставки заробітньої плати для кожного із працівників. Він здійснює збереження, редагування, а також видалення даних, що стосуються обліка робочого часу.

Працівник – той хто в основному отримує інформацію про свій тарифний оклад та відповідну йому заробітну плату, щоденно звітується за виконану роботу.

Кожен з користувачів має свої функції для роботи. Загалом система виконує наступні запити і функції:

1- вивід інформації:

а) вивід табеля обліку затраченого робочого часу для кожного із працівників;

б) вивід поточного тарифного окладу та відповідної йому заробітної плати для кожного із працівників;

в) вивід історії змін ставки заробітної плати;

г) вивід інформації про працівників;

д) вивід інформації про посади працівників;

е) вивід інформації про підрозділи підприємства;

є) вивід розрахунку заробітної плати;

2- внесення, редагування і видалення даних:

а) внесення, редагування і видалення інформації про працівників;

б) внесення, редагування і видалення інформації про посади працівників;

в) внесення, редагування і видалення інформації про підрозділи підприємства;

г) внесення, редагування і видалення інформації тарифну ставку для кожного із працівників;

д) внесення робітниками інформації про виконану роботу.

1.1.4 Опис ключових варіантів використання [17-20]

Отже, було виявлено основного актора даної системи, а саме: адміністратора. Дана система може потребувати інших додаткових акторів, але можливо обійтись і без них, адже їх ролі можуть виконувати два наявних актора.

Таблиця 1.1 – Актори системи

№	Основний актор	Найменування	Формулювання
1	Адміністратор	Коригування інформації про працівників	Дозволяє введення, редагування, а також видалення інформації про працівників.
2	Адміністратор	Коригування інформації про посади працівників	Дозволяє введення, редагування і видалення інформації про посади працівників.
3	Адміністратор	Коригування інформації про підрозділи підприємства	Дозволяє введення, редагування , а також видалення інформації про підрозділи підприємства.
4	Адміністратор	Коригування тарифної ставки ставки зарбітної плати для кожного працівника	Дозволяє введення, редагування і видалення тарифної ставки ставки зарбітної плати для кожного працівника.
5	Адміністратор	Коригування табеля обліку використаного робочого часу	Дозволяє введення, редагування , а також видалення тебеля обліку використаного робочого часу.
6	Адміністратор	Перегляд даних	Має можливість переглядати будь-яку інформацію, надану йому в його інтерфейсі
7	Робітник	Перегляд даних	Має можливість переглядати будь-яку інформацію, надану йому в його інтерфейсі
8	Робітник	Звітування за виконану роботу	Дозволяє введення інформації про відпрацьовані години.

Далі, на рисунку 1.1 я представлю основні варіанти використання системи.

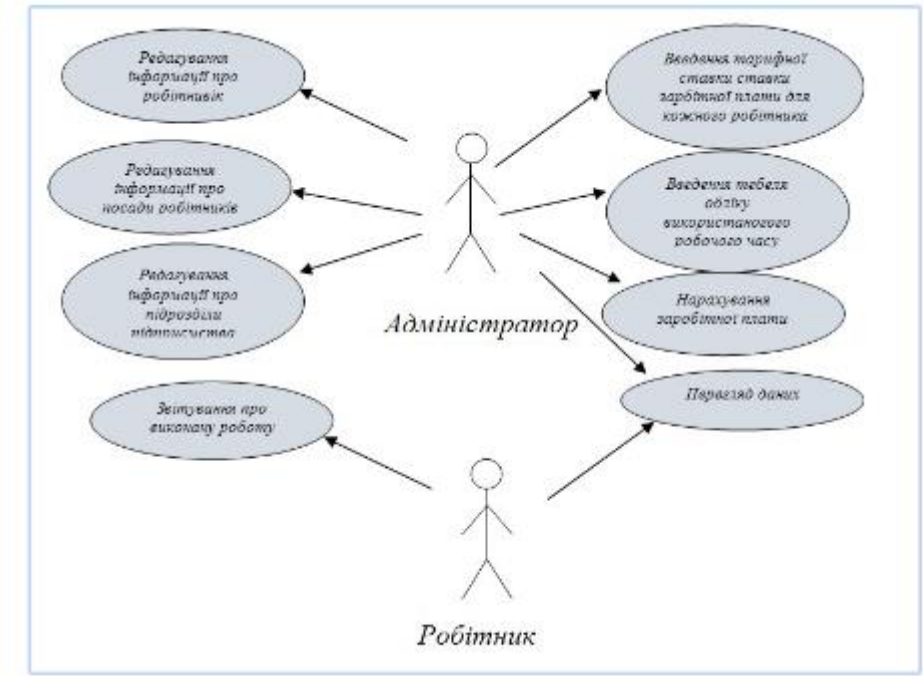

Рисунок 1.1 – UML діаграма варіантів використання системи

Таким чином, підсумовуючи, можу сказати, що впровадження розробки дасть змогу автоматизувати виконання рутинних щоденних завдань підприємства із будь-якою формою власності та кількістю працівників. При цьому отримується взаємозв'язок поміж різними службами, що надасть можливість підвищити ефективність праці і дозволить позбутися значної кількості можливих помилок. Окрім функцій керування, система надасть додаткових можливостей для підвищення рівню сервісу для користувачів. Крім цього передбачено можливість врахування різноманітних побажань і переваг кінцевих користувачів, зберігання даних по кожному користувачеві, що коли-небудь використовував можливості системи, і процес для нього стає ще більше зручним та безпроблемним.

1.2 Проєктування програмної системи [21-34]

Якщо говорити про те, як розміщати і, головне, де інструменти розробки, то найдоцільніше це робити на сайті із різним правами доступу до різних інформаційних блоків, зокрема ним зручно користуватись, оскільки він, в цілому, є фактично набором сторінок, на яких можуть міститись текстові частини, картинки, а також мультимедійні чи ж об'єкти іншого плану. З метою підвищення зручності при перебуванні відвідувача так само як і для полегшення при пошуку необхідної інформації, важливо передбачити чітку та логічно продуману структуру.

Структурою можна назвати внутрішній устрій, його основу, розміщення сторінок, структурних одиниць, таких як: розділи, підрозділи, додаткові матеріали. На мою думку, першочергово є перетворити хаотичний набір інформації в структурований та зрозумілий, з який вже буде зручно користуватись та працювати користувачеві.

Із позиції розробника зазначу, що умовно структуру можна розділити на два рівні, один із яких є логічним (тут є сукупність сторінок, що об'єднані поміж собою як єдиним дизайном, так і стилем та покликаннями), інший фізичним (тут йдеться про впорядкований набір файлів різних за типом, наприклад HTML-сторінки, зображення і програми, можливо мультимедійні файли).

Таким чином, якщо вже спочатку добре продумати файлову структуру, то це дасть змогу мені, як розробникові, оптимізації своєї роботи і, що важливо для роботи в команді, буде зрозумілим для інших фахівців, які працюють чи працюватимуть над проєктом.

1.2.1 Логіка структури

З попереднього досвіду я розумію, що розробки із недопрацьованою логікою в структурі ускладнюють роботу відвідувачів та користувачів. Перш за все, завдяки вдалому орієнтованому на цільову аудиторію та користувача дизайну, можна легко знайти і використати будь-яку інформацію, яку на ньому розміщено. При цьому абсолютно зрозумілими повинні бути як сама логічна схема так і логічність взаємозв'язку поміж окремими сторінками.

Хороша навігація означатиме, що користувачі будуть точно розуміти де саме і для чого вони на ресурсі, де розташовані та для чого призначені і як використовувати ці елементи. Правильна структура інформації дозволяє легко користуватися навігацією та бути впевненими користувачами і завжди повернатися до раніше переглянутих сторінок.Саме це і є основним моїм завданням як розробниці.

Таким чином, логічне проєктування передбачатиме організацію інформації, побудову самої структури та навігації по роздільно.

Важливо в процесі проєктування інформаційного наповнення та засобів навігації дотримуватись чотирьох основних принципів, які ґрунтуються на сприйнятті інформації людиною, зокрема:

1- використання позначень;

2- доцільність;

3- однотипність;

4- поділ на частини.

Важливим моментом є застосування усталених і загальноприйнятих слів і термінології. Саме в цьому сенсі вважаю не дуже вдалим використання піктограм, оскільки буває, що відвідувачі можуть неоднозначно їх тлумачити, таким чином спотворюючи вміст. Саме тому це варта розміщенні змістовних елементів, посилань та елементів навігації разом із їх позначеннями, уникати не однозначності та невиразності термінів на сторінці.

В розділах потрібно надавати детальні інформацію, елементи, що відповідають суто цьому розділу або ж фрагменту цілої сторінки, стосовно невідповідних елементів, то їх за краще перенести чи ж взагалі прибрати.

Варта також передбачити щоби подібні, одинакові елементи сторінок мали одинаковий розмір і були в одному і такому ж самому стилі. Якщо ж потрібно щоби сторінки таки мали відмінності, користувачу потрібно чітко розуміти доцільність.

Врахувала я також і психологічні особливості людини, однією з яких є те, що ми не можемо одночасно зберігати великі об'єми інформації. Зазвичай, людям достатньо легко сприйняти інформацію, у якій міститься від 4-ох і до 6-ти різних елементів і саме тому, користувач буде краще орієнтуватись та швидше знаходити потрібні для нього матеріали, особливо якщо вони структуровані та розподілені на групи, зокрема про це йдеться і в працях мого керівника магістерської роботи, к.т.н., доц. Цуприк Г.Б., [4,10].

Враховуючи вище сказане зазначу, що саме така організація (саме як впорядкована структура) повинна інформувати користувачів, що саме він може знайти де саме шукати. Структура повинна нагадувати зміст книги: якщо його добре пропрацьовано та викладено, то одразу зрозуміло, що варта очікувати від самої книги та її вмісту. Інформацію потрібно організувати таким чином, щоб користувач чітко розумів, що на нього чекає на наступній сторінці лише за назвою посилання, яке його зацікавило.

Таким чином, опрацювавши багато джерел і ознайомившись із подібними розробками, мені потрібно обрати одні із видів організації інформації, зокрема посеред лінійної, ієрархічної, мережевої (контекстно-залежної), комбінованої.

Для лінійної структури сторінки характерним є послідовний перегляд сторінок чи вкладок. Проте, зазвичай у «чистому вигляді» таку структуру застосовувати не доцільно, оскільки це не зручно при використанні. Частіше, такий підхід застосовний для поділу достатньо довгої сторінки на декілька коротших проте обов’язково логічно- відоокремлених частин.

Для ієрархічної структури характерним є наявність головної сторінки, з розміщенням на ній меню із покликаннями на розділи, які розташовані на наступних сторінках. Таким чином в розділах можуть міститися покликання як на підрозділи так і на іншу детальну інформацію.

Найбільш поширеною є мережева структура. За такої організації можна переходити від сторінки до сторінки, яка з нею логічно взаємопов'язана. Для такої структури притаманно, що покликання на інші розділи формуються в залежності від логіки користувача. Хорошим прикладом структури такого типу є великі портали інформаційного спрямування.

Залежно від задуму розробника не обов'язковою є прив'язка між собою усіх сторінок, проте взаємозв'язки повинні бути, і бути максимально природніми і добре відображати саму логіку.

Комбінована структура зручна тим, що в ній поєднується кілька структур. Для прикладу, основною може бути ієрархічна структура, але при необхідності у певному місці може міститися лінійна структура, при якій виконання дій відбувається покроково.

Також для мене було важливо врахувати думку та побажання кінцевого користувача системи. При створенні естетика і краса для мене на останньому місці, перевагу я надавала релевантності, зручності пошуку та відповідності вимогам. При розробці я не покладалась на інтуїтивністт, оскільки вважаю, що структура може бути очевидною для мене як розробника, проте, але абсолютно не зрозумілою для користувачів, а це не прийнятно.

1.2.2 Опис фізичної структури

Перед тим як писати програму, для початку поясню, що я робила на прикладі системи, в якій передбачено лише HTML-сторінки. Спочатку я

створила та назвала кореневу папку на локальному диску, назвала за майбутньою доменною адресою, як наприклад, site.ua.

Усі файли, які стосуватимуться системи варта розмістити в кореневій папці. Але, перш за все я розробила структуру: які будуть та як називатимуться розділи та підрозділи. Для прикладу наведу типову структуру моєї сторінки.

1- головна сторінка;

2- меню.

Таким чином кожен розділ є як окрема сторінка. Перед тим, як присвоїти імена файлам, потрібно врахувати певні налаштування веб-серверу, під керуванням якого і буде працювати система, тобто йдеться про наступні моменти:

1- який файл буде завантажуватись як перший

2- чи можливо закрити доступ до певних окремих файлів

3- як та яким чином можна відстежувати помилки

4- інше.

Провівши детальний аналіз я прийняла рішення при розробці своєї програмної системи реалізацію здійснювати використовуючи деревовидну структуру. Таким чином в мене наявна головна сторінка підприємства на якій є навігаційна панель та форма авторизації. Для користувачів передбачено деякі певні розділи на які вони мають можливість переходити. Кожний розділ відрізнятиметься та складатиметься з інших сторінок, але покликання є на будь-які потрібні сторінки при наявності доступу. Можливість обирати надається навігаційною панеллю. Саме така структура, на мою думку, є оптимально зручною, оскільки сторінки погруповані та поділені на розділи. Розроблювана систему не міститиме надмірної кількості сторінок одного рівню. Поки по структурі все, але далі я ще повернуся до неї.

Об’єктна модель документу (на англійській це Document ObjectModel або DOM) являється стандартом, яким регламентується спосіб представлення вмісту документу, зокрема веб-сторінки (тобто йдеться про все те, що може

розміщуватись на сторінці типу веб, а саме: малюнки, покликання, тексти та подібне) в вигляді набора різних об’єктів [35,36].

Важливим моментом є й те що, відмінністю від DOM – об’єктної моделі браузеру, що являється унікальною для кожного браузеру, об’єктну модель документу прийнято вважати стандартом та такою, яку повинні підтримувати усі без виключення браузери. І хоч практично підтримка об’єктної моделі браузеру була реалізована не повною мірою, однак варта максимально можливо усім дотримуватись вимог цього стандарту.

Об’єктна модель зазвичай розбита на 3-и рівня. Першим рівнем є перша версія стандарту і на сьогодні єдина завершена. Він складається із 2-х розділів: першим є ядро, яким визначаються принципи для маніпуляцій зі структурою документа (ґенерація та навіґація), а другим забезпечується подання в об’єктну модель елементів HTML, які визначаються одноіменними тегами.

Щодо другого та третього рівнів, то ними описується модель подій, доповнюються таблиці стилів, проходи по самій структурі.

При об’єктній моделі документ представляють у вигляді дерева, а така структура є найбільш затребувана в програмуванні, оскільки забезпечується уніфікований варіант навігації по документові.

Візуальне ж уявлення про об’єктну модель подекуди відрізняється від HTML-кода в якому часом трапляються помилки. Тоді браузер може самостійно їх виправляти.

1.2.3 Моделювання архітектури системи

Шаблон по типу ModelView-ViewModel (MVVM) зазвичай призначений для застосування при проєктуванні архітектури додатка [37].

MVVM шаблон використовують для того щоби розділити модель з її уявленням про те, що необхідно для змін по ним розділене окремо одне від

одного. І таким чином, для прикладу, розробником задається логіка присутня при роботі із даними, а вже дизайнером проводиться робота із інтерфейсом.

У шаблонах проєктування по типу MVC/ MVP безпосередньо будь-які зміни в інтерфейсі зазвичай не впливають ніяким чином на саму Модель, а, попередньо, проходять через Controller (з англійської Контролер) чи Presenter. У технологіях типу як WPF та Silver light передбачено концепція «в'язки данних», яка дає змогу пов'язувати данні по обидві сторони із візуальними елементами. Таким чином, на мою думку, завдячуючи використанню цього прийому використання моделі MVC стає надто незручним саме через прив'язку інформаційних даних до подачі, а це напряму не вкладається до концепції такого типу (йдеться про MVC / MVP).

Шаблоном ModelView-ViewModel передбачено 3-и основні компоненти і до них відноситься модель, яка представляє собою бізнес-логіку самої програми; представлення користувача інтерфейса XAML, а також модель, в котрій є уся логіка побудови графічного інтерфейсу та покликання на саму модель, і саме по цій причині він представляється як модель для представлення.

Існують ознаки за якими можна говорити про відповідність View-моделі. До них можна віднести:

1- двосторонню комунікацію з поданням;

2- View-модель як абстракцію уявлення, що найчастіше може означати, що властивості уявлення збігатимуться із властивостями моделі по типу View / моделі;

3- для View-моделі не існує покликання на інтерфейс подання (I View). При зміні стану View-моделі автоматично змінюється уявлення і навпаки, так як діють механізми призначенням яких є скріплення даних(Bindings);

4- один примірників View-моделі, пов'язаний із одним із відображень.

Таким чином при застосуванні такого патерну, поданням не реалізовується відповідний інтерфейс (IView). Саме ж Подання повинне мати покликання на джерело даних (Data Contex), і таким в нашому випадку і буде

модель по типу View. Для елементів уявлення характерно наявність зв’язку (Bind) із відповідними їм властивостями та подіями View. А Модель, вже в свою чергу, реалізовує спеціальний інтерфейс, призначенням якого є застосування з метою автоматичного оновлювання елементів для Подання.

Представлення є графічним інтерфейсом, наприклад вікно, кнопка. У тому випадку, коли у Моделі представлення змінюється якась із властивостей, то саме Представлення, в свою чергу, робить запит на оновлене значення властивості. У випадку, якщо користувачем здійснюється вплив на якийсь із елементів інтерфейсу, Представленням викликається відповідна команда, яка надається Моделлю уявлення.

Таким чином Модель представлення, з однієї сторони, як абстракцією Представлення, однак якщо подивитися з іншого боку, нею надається обгортка данних із Моделі, які необхідно буде скріплювати. Підсумовуючи вище сказане, в ній міститься Модель, яку перетворено до Представлення, та команди, за допомогою яких Представлення може використуватися з метою впливу на Модель. Зазвичай, застосовується у вигляді шаблонів, в яких містяться методи призначені для забезпечення роботи в цілому системи.

Таким чином, якщо говорити про архітектуру програмної системи, то вона, головним чином, визначає її складові рівні. В моєму випадку архітектура представлена на рисунку 1.7, вона є оптимальною та достатньо простою, і до її складу входять три рівні.

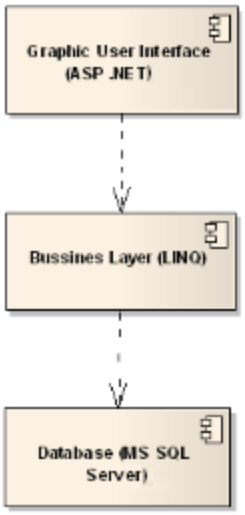

Рисунок 1.7 – Архітектура Моделі розроблюваної системи

1) Graphic User Interface – графічний інтерфейс користувача, представ ле ний у вигляді Web-форм ASP .NET;

2) Bussines Layer – рівень бізнес-логіки системи, реалізований з використанням технології LINQ, на цьому рівні головним чином здійснюється обробка даних, що читаються та записуються у базу даних;

3) Database – рівень бази даних (на основі MS SQL Server), який забезпечує зберігання даних системи.

Структура програмного проекту відповідно до представленої вище архітектурної моделі також складатиметься із 3-х компонентів, що значною мірою розширює можливості масштабованості системи та спрощує внесення змін в програмний код.

1.2.4 Вибір системи керування БД [38]

В загальному базою даних – називають саме впорядкований набір логічно пов’язаних даних, які використовуються спільно різними користувачами за їх потребами, які зберігаються разом і в одному місці. Найпростішою базою для

збереження інформації є табличне представлення даних з традиційними рядками та стовпцями.

З досвіду використання бази даних можна можу виділити їх притаманні робочі характеристики, до яких варта віднести:

1- повноту (без надмірності проте достатній розмір щоб точно знайти потрібну інформацію);

2- правильну організацію (оптимальна структура дозволить достатньо легко віднайти потрібну інформацію);

3- актуальність (оновлення є важливою умовою для того щоби вважати базу даних точною та повною);

4- зручність при використанні (простота та зручність при використанні разом із розвиненими методами доступу до частини чи всієї інформації);

Ну і базами потрібно керувати при чому системно. Для цього існує сукупність програмних та лінгвістичних засобів як з загального спрямування так і спеціально призначених, які і забезпечують керування їх створенням та використання.

До основних функцій системи для керування можна віднести:

1- управління даними на дисках (мова про зовнішню пам’ять);

2- використання дискового кешу для управління даними оперативної пам’яті;

3- можливість відновнення після збою в роботі, для цього потрібно фіксувати зміни, здійснювати резервне копіювання, відновлювати;

4- підтримку мови визначення даних та мови маніпулювання даними.

До компонент бази даних відносяться:

1- ядро, основним призначенням якого є керування даними (фіксування змін, зовнішня та оперативна пам’яті);

2- процесор мови яким забезпечується процес оптимізації запитів на вилучення та зміна даних і створення внутрішнього коду;

3- підсистема призначенням якої є підтримка виконання, якою інтерпретуються програми для маніпуляції даними, які створюють

користувальницький інтерфейс з системи керування;

4- сервісні програми (так звані зовнішні утиліти), якими забезпечуються цілий ряд додаткових можливостей для обслуговування інформаційної системи.

Класифікуються системи керування за моделями даних (ієрархічну, мережеву, реляційну,об'єктно-орієнтовану та об'єктно-реляційну); за ступенем розподільності (локальна, коли всі її частини розміщують на одному комп'ютері; розподілена, коли частини системи керування можуть розміщуватись на 2-ох і більше комп'ютерах); та за способом доступу до бази данних (файл-серверна).

У файл-серверних СУБД-их характерно, що файли даних розташовані на файл-сервері централізовано. Систему керування розташовують на кожному із комп'ютерів клієнта (робоча станція), а сам доступ системи до даних здійснюється вже з використанням локальної мережі. Синхронізація читань як і оновлення здійснюють при використанні файлових блокувань. До переваг цьої архітектури варта віднести її низьку навантаженість на процесорі файлового серверу. Суттєвим недоліком, із-за якого така технологія та її використання і цілому вважається вже застарілою є потенційно високе завантаження на локальну мережу; утруднене або й неможливе централізоване керування, неможливість забезпечити високу надійність, високу доступність та високу безпеку. Про те її все ще використовують, зокрема для локальних додатків, в яких використовуються функції керування базами даних а також для систем з низькою інтенсивністю при обробці даних та достатньо низьким рівнем пікових навантажень на саму базу даних.

Клієнт-серверна СУБД-их розташована на сервері заразом із базою даних і безпосередньо до неї здійснюється доступ. Стосовно усіх клієнтських запитів, що мають відношення до опрацювання даних, то вони централізовано виконуються клієнт-серверною СУБД-их. До нНедоліків клієнт-серверних СУБД-их варта віднести підвищені вимоги до сторони серверу. Щодо переваг, то тут слід звернути увагу на ймовірно більше низьке завантаження на локальну

мережу та в цілому вона є зручною з погляду централізованого керування та при забезпеченні високої надійності, доступності та високого рівня безпеки.

Вбудованою СУБД-их називають такий їх тип, що поставляється в якості складової частини для конкретного програмного продукту при цьому не потребуючи проведення самостійної установки. ЇЇ призначено для локального зберігання даних і вона не розрахована для колективного використання у мережі. Фізично вбудовану СУБД-их часто реалізують в вигляді підключення до бібліотеки. Сам доступ до даних зі сторони додатка може реалізовуватись можливостями SQL чи засобами спеціальних програмних інтерфейсів.

Система керування за принципом безпосереднього запису підтримує стратегію при якій усі змінені блоки даних відразу записуються до зовнішньої пам’яті після отримання сигналу підтверджуючого будь-яку транзакцію. Однак таку стратегію можна використовувати лише за високої ефективності для зовнішньої пам’яті.

Система керування за принципом відкладеного запису передбачає акумуляцію змін в буферах для зовнішньої пам’яті ад до того моменту поки не настане одна із подій, зокрема:

1- контрольна точка;

2- мало простору зовнішньої пам’яті, відведеного для журналу. Системою керування виконується контрольна точка і запис журналу розпочинається спочатку при цьому попередня інформація затирається;

3- зупинка. Під час настання цієї події система керування очікує, коли весь вміст усіх буферів зовнішньої пам’яті будуть перенесені до зовнішньої пам’яті, і вже після цього фіксується, що дія виконана коректно.

Тепер про неструктурований підхід (англійською скорочено NoSQL), який спочатку вважався як варіант реляційної системи керування, однак з різницею яка полягала в тому, що уникалось використання стандартів типу SQL. Вже далі ця ідея виявилась вдалою, то почала використовуватись і у випадках з базами даних не реляційного типу.

Задум був таким, що при неструктурованому підході NoSQL база даних і СУБД не передбачають внутрішніх зв’язків. Тут не передбачено одну модуль оскільки кожною із баз даних в залежності від кінцевої мети використовуються різні моделі.

Перевагою неструкктурованого підходу є те, що тут невілюються усі обмеження притаманні реляційній моделі, зокрема це стосується її недостатньої продуктивності, трудомісткого горизонтального масштабування, недостатньої продуктивності в кластері), також значно простіше та легше відбувається процес зберігання та доступу до даних в цілому. При використанні в базах даних неструктурованого підходу знімаються обмеження щодо жорсткості зв’язків і пропонуються різні типи для до специфічних даних.

Такод, що відмінне від баз даних з реляційним підходом, є можливість групування колекцій даних з іншими базами даних з неструктурованим підходом, як наприклад з MongoDB. В таких та подібних Системах керування дані в базі зберігаються як єдине ціле. Разом з тим дані такого типу можуть представлятися в якості одиночного об’єкту (як JSON) і одночасно виконувати коректні відповіді на запити до полів.

В базах даних з неструктуризованим підходом зазвичай не використовується загальний формат запиту, як це здійснюється в SQL у базах даних реляційного типу. В кожному NoSQL рішенні використовується своя власна система запитів.

До прикладів використання NoSQL варта віднести:

1- вкладену інформацію, оскільки відмічено, що сховища з документо-орієнтованим підходом відмінно себе зарекомендували при роботі з глибоко вкладеною та достатньо складною інформацією;

2- підтримку JavaScript, оскільки однією із особливостей яка і відрізняє сховища за документо-орієнтованим типом є те як вони співпрацюють з іншими додатками, зокремаJSON;

3- швидкодія, оскільки для таких баз даних вона зазвичай значно вища;

4- безсхемну розробка, оскільки при використанні NoSQL пропонуються гнучкіші рішення, на відміну від реляційного підходу де вимагається чітко описана структура даних ще до початку роботи.

5- автоматизовану (чи спрощену) реплікацію (масштабування). Для NoSQL тенденція швидкого розвитку, тому розробники зацікавленні більш активно вирішувати основні проблеми, однією з яких є реплікація та масштабування. При NoSQL підході рішення легко масштабувати та працювати з кластерами, чого не скажеш про реляційний підхід;

6- широкий вибір різних моделей, коли повстає питання обрати NoSQL сховище

В результаті аналізу я ознайомилась із різними моделями та функціональними системами для баз даних типу NoSQL. Враховуючи специфіку розроблюваної системи для своєї роботи я надала перевагу документно-орієнтованій базу даних, яка є більш гнучкою та швидкодійною, ну і ще ця база даних є простішою за аналоги. Таким чином в моїй роботі я використала MS SQL Server.

1.2.5 Побудова схеми бази даних

MS SQL Server [39] є системою управління базами даних реляційного типу, до особливостуй якої варта віднести:

1- можливість підтримки різних платформ;

2- високу продуктивність;

3- високий ступінь захисту даних;

4- наявність потужних засобів роботи із даними;

5- можливість зберігати великі масиви даних;

6- можливість зберігати данні, які передбачають дотримання режиму конфіденційності чи секретності або повного недопущення їх втрати.

Основною особливістю розроблюваної бази даних є створети таблиці і зв’язки між ними. Саме такий підхід дозволить створити таблиці та наповнювати їх даними структуровано.

До переваг моделі реляційного типу можна віднести:

1- простоту і доступність при розумінні користувачем, оскільки єдиною інформаційною конструкцією, яка використовується є конструкція табличного типу;

2- жорсткі вимоги до проєктування, які, проте, ґрунтуються на відомому математичному апараті;

3- повна незалежність даних, тобто зі зміною прикладної програмі зміни в реляційній базі даних є мінімальними;

4- при організації запитів та написання прикладних програм не потрібно знати конкретику щодо організації бази даних зовнішньої пам’яті.

Проте реляційна модель має і свої недоліки, зокрема:

1- не кожну предметну область зручно і оптимально представляти у табличні формі;

2- результатом логічного проєктування є множинна сукупність «таблиць», і це може призвести до труднощів в розумінні структури даних;

3- для бази даних потрібен відносно не великий об’єм вільного місця на зовнішній пам’яті;

Таким чином в розробленій базі даних містяться такі таблиці передметної області, докрема:

Department (Підрозділ виробничої служби), Post (Посада (фах) працівника), Employee (Працівник), Salary Mistory (Тарифний оклад заробітної плати), Work Log (Табель обліку використання робочого часу).

Ну і на завершення, під час розробки баз даних, між таблицями використовують наступні зв’язки, а саме:

-один-до-одного - одній сутності у однозначну відповідність ставиться лише одна але інша сутність;

-один-до-багатьох (чи багато-до-одного), коли одній сутності у відповідність ставиться кілька інших сутностей;

-багато-до-багатьох, в зв'язку із можливою не однозначність реалізовувати такий тип взаємозв'язку не доцільно, наслідком впровадження такої реалізації може бути порушення цілісності в описі предметної області, що є критично. Якщо ж все таки необхідно реалізовувати зв'язок такого типу, то доцільно це зробити через два зв'язки, а саме багато-до-одного та один-до-багатьох.

На рисунку 1.12 представлено використані між табличні зв'язки.

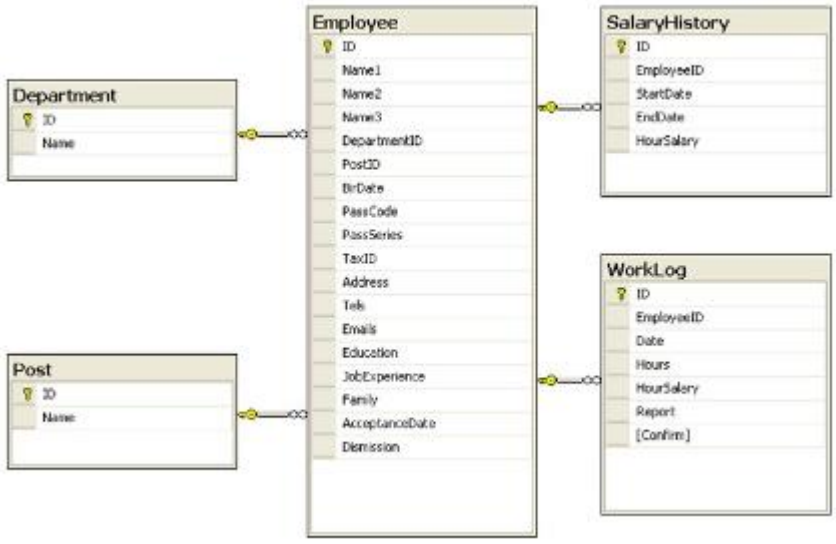

Рисунок 1.12 – ER діаграма бази даних

Підсумовуючи, в результаті аналізу та враховуючи доцільність для встановлення взаємозв'язку поміж таблицями я скористувалась зовнішнім ключом, тобто вторинний ключ, який представляє собою одне поле (чи декілька полів чи стовбців) в таблиці, які містять покликання на поле (поля) первинного ключа який розміщений в іншій таблиці. Саме зовнішнім ключом і визначається спосіб за яким об'єднуються таблиці між собою. Таким чином буде дві логічнопов'язані між собою таблиці : «таблиця первинного ключа» яка ще може називатись головною таблицею, та «таблиця вторинного (зовнішнього)ключа» або яка ще може називатись підпорядкованою таблицею.

1.3 Конструювання програмної системи [40-43]

1.3.1 Обґрунтування вибору технологій розробки

Microsoft. NET Framework є тією платформою яку я використала для своєї розробки та яка вважається базовою та такою за допомогою якої можна об’єднати потрібні для процесу розробки технології та інструментальні засоби. Мій вибір як базис з метою реалізації представленого мною програмного проєкту було зумовлено зокрема й тим, що цією платформою об’єднується цілий комплекс зручних та достатньо потужних засобів веб-програмування, обробка даних, а я як розробник отримала потужний сервер баз даних (MS SQL Server), та можливість всецілої підтримки інтегрованого середовища розробки MS Visual Studio, за допомогою якого я зможу реалізувати практично усі задачі на етапі кодування.

Microsoft .NET Framework – розвинена платформа розробки, розгортання та виконання розподілених програм. Центральною частиною каркасу .NET Framework є виконуване середовище CLR (CommonLanguage Runtime), або .NET Runtime. Код, що виконується під управлінням CLR часто називають керованим кодом (IL або MSIL). Проміжна мова MSIL схожа на байт-код Java, оскільки в її основі лежить ідея низькорівневого представлення мови у вигляді числових кодів, що можуть бути легко трансльовані у машинні коди, специфічні для платформи.

Наявність IL розкрило перед .NET широкі можливості, зокрема

1) кросплатформеність;

2) можливості взаємодії різних мов програмування, що підтримуються в .NET, в рамках одного проекту;

3) введення спільної системи типів (CST – Common System Types), що описує типи даних, які доступні в IL, тому всі мови програмування, орієнтовані на .NET, генерують код, який базується на єдиний системи типів.

Серйозним аргументом на користь .NET є технологія можливість керувати пам’яттю, котра базується на принципі «збирання сміття». Головною ідеєю сього під ходу є те, що якщо об’єкт не використовується (тобто кількість посилань на нього рівна нулю), то його слід знищити. Такий підхід знімає з розробника відповідальність за звільнення зарезервованої динамічної пам’яті і, як наслідок, значно спростити безпосередньо сам процес розробки.

Для встановлення середовища .NET на ПК необхідно отримати (безоплатно) його дистрибутив з офіційного сайту Microsoft.

1.3.2 Вибір мови програмування

В якості основної мови програмування при реалізації представленого на розгляд екзаменаційної комісії програмного проєкту я обрала мову C# [44,45].

Офіційно Microsoft представляє C# в якості простої, сучасної, об’єктно-орієнтованої і безпечної щодо типів мову програмування, наслідувану від мов C/C++.

C#, на сьогоднішній день, є однією із найпотужніших об’єктно-орієнтовних мов програмування. Крім того, ця мова повністю орієнтована на платформу .NET Framework, що забезпечує ефективне використання максимальної кількості можливостей останньої. Мова C# також досить зручна в програмуванні, що в поєднанні з інструментальними засобами середовища VisualStudio, робить процес розробки максимально «приємним» для програміста.

C# – відносно нова мова програмування, що характеризується наступними основними можливостями:

– повною підтримкою програмування з об’єктно-орієнтованим підходом;

– чітко визначений набір базових типів;

– вбудованою підтримкою автоматичного генерування XML-документації;

– автоматичним вивільненням виділеної динамічної пам'яті;

– можливості маркування властивостей і методів атрибутами;

– повна підтримка бібліотеки класів .NET;

– можливістю доступу напряму до пам’яті через вказівники;

– підтримка властивостей та подій;

– можливості використання для написання динамічних сторінок ASP.NET та Web-служб XML.

Варто зауважити, що хоча й C# призначена для розробки коду, що виконується середовищем .NET, сама вона не є частиною .NET. Існують деякі можливості, які підтримуються .NET, але не підтримуються C#. І, як не дивно, існують можливості C#, які не підтримуються .NET.

1.3.3 Обґрунтування вибору технологічного засобу реалізації веб-проєкту

Основним технологічним засобом реалізації представленого веб-проєкту є платформа ASP.NET [46]. З її допомогою розроблено клієнтська (Web-форми) та серверна частина програми. В цьому пункті представленого основні можливості, які надала мені, як розробниці, ця технологія.

В середині 90-х років минулого століття Тімом Бернерсом-Лі (Tim Berners-Lee) була здійснена перша передача файлів за допомогою HTTP (HypertextTransferProtocol – в перекладі з англійської це протокол передачі гіпертекстових файлів). З того часу популярність HTTP виросла настільки, що сьогодні це майже побутовий термін [47].

Версія ASP. NET 1.0 стала першим кроком компанії Microsoft до створення комплексного набору компонентів, який поєднує стандартні блоки Web (мови розмітки та HTTP) з потужною об’єктно-орієнтованою методологією.

Між платформою ASP.NET і більш ранніми платформами Web-розробки визначають наступні відмінності:

– ASP.NET пропонує повністю об’єктно-орієнтовану модель програму ван ня, що включає подієорієнтовану компонентну структуру, яка забезпечує інкапсуляцію та повторне використання програмного коду;

– ASP.NET дозволяє писати код на будь-якій мові, що підтримує .NET (наприклад, C#, VB.NET, J# та ін.);

– ASP.NET також є платформою для побудови Web-служб, що представляють собою програмні компоненти, які викликаються іншими програмами в рамках різних платформ;

– ASP.NET орієнтована на досягнення високих показників продуктив ності – сторінки та компоненти ASP.NET компілюються за вимогою, а не інтерпретуються при кожному використанні.

Про ASP.NET важливо знати наступні факти:

1) повна інтеграція з .NET Framework;

2) компільованість, а не інтерпретованість;

3) підтримка кількох мов програмування;

4) функціонування в рамках середовища CLR (CommonLanguage Runtime);

5) об’єктна орієнтація;

6) підтримка великої кількості пристроїв та браузерів;

7) легкість розгортання та конфігурування.

1.3.4 Механізм опрацювання даних та доступ до бази даних

В представленому мною програмному проекті в якості механізму обробки даних та доступом до бази данних я обрала LINQ (зокрема, LINQ to SQL).

Такий вибір зумовлено тим, що ця, порівняно молода технологія, надає ряд гнучких та зрозумілих механізмів роботи з даними.

В процесі розвитку платформи .NET і підтримуваних нею мов C# та VB, стало зрозуміло, що однією із найбільше проблемних областей для розробників залишається доступ до даних із різноманітних джерел. Маніпуляція з XML та безпосередній доступ до бази данних часто, в кращому випадку, заплутані, а в гіршому взагалі проблемні.

Проблеми баз даних різноманітні. По-перше, складність представляє те, що не можна програмно взаємодіяти із базою даних на рівні звичайної мови. Це призводить до синтаксичних помилок, які не можна виявити до моменту запуску. Неправильні посилання на поля бази даних також не виявляються. Це може погано відобразитися на програмі, особливо якщо це відбудеться в момент виконання коду обробки помилки. Наявність коду бази даних, який не перевіряємо при здійсненні компіляції, може призвести до цієї проблеми.

Друга проблема пов’язана із незручністю, яка викликана різними типами даних, що використовуються певним доменом даних, наприклад, різниця між типами бази даних або типами даних XML і звичайною мовою програмування.

Аналіз, ітерація і маніпуляція XML можуть бути досить обтяжливими. Часто фрагмент XML – це все, що необхідно, але через W3C DOM XML API об'єкт XmlDocument повинен бути створений тільки для того, що виконувати операції лише із фрагментом XML.

Замість того щоб додати більше класів і методів для поступового усунення таких недоліків, командою розробників Microsoft було прийняте рішення піти на один крок далі по питанню абстрагування основ для запитів даних із цих конкретних доменів даних. В результаті цього і виникла LINQ, як одна із технологій Microsoft, призначена для забезпечення механізму рівня мови, що підтримує запити даних багатьох типів. Ці типи включають масиви та колекції в пам’яті, також бази даних, документи XML тощо.

Головним чином LINQ орієнтована за запити, зокрема ті, що повертають набір об’єктів, одиночний об’єкт чи підмножину полів із об’єктів. В LINQ

набір, що повертається, називають послідовність (sequence). Типом більшості послідовностей LINQ є IEnumerable<T>, в якому T є типом даних об'єктів, які перебувають в послідовності.

До MS .NET Framework-у входять ряд механізмів LINQ, кожен з яких орієнтований на роботи із специфічним типом доменів даних.LINQ to Objectcs – назва дана IEnumerable<T> API для стандартних запитів (StandardQueryOperators). Саме LINQ to Objectcs дозволяє виконувати нам запити до масивів і колекцій даних, які знаходяться в пам'яті.

- LINQ to Objectcs – назва дана IEnumerable<T> API для стандартних запитів (StandardQueryOperators). Саме LINQ to Objectcs дозволяє виконувати нам запити до масивів і колекцій даних, які розміщені в пам'яті.
- LINQ to XML – назва дана XML API, що призначений для роботи з XML. Microsoft не лише додала необхідні бібліотеки XML для роботи з LINQ, але також усунула недоліки стандартної моделі XML DOM, що значно полегшило роботу з XML.
- LINQ to DataSet – назва дана програмному інтерфейсу LINQ API для DataSet. На даний час існує ряд проектів, що опираються на DataSet, які завдяки даній можливості можуть скористатися механізмом LINQ.
- LINQ to SQL – назва, дана програмному інтерфейсу IQueryable<T>, який дозволяє запитам LINQ працювати з базою даних SQL Server-у.
- LINQ to Entities – альтернативний інтерфейс LINQ API, що використовується для звернення до бази даних. Він визначає сутнісну об'єктну модель від фізичної бази даних, вводячи логічне відображення між ними двома. З таким поділом зростає потужність та гнучкість, але, заразом із тим, зростає і складність.

Може скластися думка, що LINQ – це механізм, пов'язаний лише із запитами, оскільки розшифровується як «мова інтеґрованих запитів» (LanguageIntegratedQuery). Однак не слід думати про LINQ лише в цьому контексті. Можливо, більш вдалою назвою для LINQ була б назва «механізм

ітерації даних» (Data Iteration Engine), але, напевно, Microsoft не захотіла називати технологію DIE (прямий переклад якої би звучала як померти).

1.3.5 Обґрунтування вибору середовища розробки

В якості головного інструментального засобу для реалізації розроблюваного програмного проєкту я обрала інтегроване середовище розробки MS Visual Studio. Таке рішення було зумовлене гнучкістю та потужністю можливостей цієї системи, за допомогою якої можна суттєво підвищити як ефективність роботи розробника так і якість в цілому програмниго виробу, та його повну інтеграцію з . NET Framework.

До основних можливостей MS Visual Studio відносять [49,50]:

1. WYSIWYG («що бачимо, те й отримуємо»). Такий підхід може бути використанодля візуально розробки Web-форм ASP . NET, так само як і для графічних інтерфейсів, що базуються на WPF (Windows Presentation Fundation).

2. Написання мінімальної кількості коду розробником. Ця можливість передбачає автоматичну генерацію стандартного програмного коду (наприклад, для Web-форм ASP.NET).

3. Інтуїтивно зрозумілий стиль кодування. За замовчування Visual Studio форматує код розробника безпосередньо під час його набору, що забезпечує формування зручної структури коду.

4. Вбудований Web-сервер. Для хостінгу програми ASP . NET необхідне програмне забезпечення, схоже на Internet Information Services. Завдяки вбудованому Web-серверу програму можна запускати безпосередньо із середовища розробки.

5. Розробка на кількох мовах програмування. Visual Studio дозволяє кодування на будь-якій мові з використання однакового інтерфейсу. Більше

того, Visual Studio дозволяє додавати Web-сторінки, реалізовані на C#, до проекту, в якому Web-сторінки реалізовані з використанням Visual Basic.

6. Підвищення швидкості розробки. Більшість можливостей Visual Studio спрямовані на пришвидшення процесу розробки.

7. Відлагодження. Система відлагодження Visual Studio дозволяє швидко виявити та виправити програмні помилки. Особливо цікавим є механізм відлагодження процесів, що запущені із операційної системи, а не із середовища Visual Studio.

У випадку представленого проекту у Visual Studio я, головним чином, здійснювала розробку ASP.NET програми (Web-програми). Мені особливо корисними були такі можливості середовища, як візуальне конструювання ASP.NET сторінок, вбудований Web-сервер, можливість автоматичної генерації коду.

1.3.6 Функціональні можливості графічного інтерфейсу користувача

Користувачі представленої системи взаємодіятимуть з нею через Web-форми. В цьому підрозділі я коротко описала функціональні можливості використаних основних Web-форм.

1. Форма «Організаційна структура» призначена для перегляду та редагування даних про підрозділи виробничої служби монтаж ного управління. Оскільки кожен підрозділ визначається лише назвою, то редагування відповідних записів здійснюватиметься безпосередньо в таблиці, де представлено їх повний перелік.

2. Форма «Посади» призначена для перегляду та редагування даних про можливі посади робітників монтаж ного управління. Оскільки кожна посада визначається лише назвою, то редагування відповідних записів здійснюватиметься безпосередньо в таблиці, де представлено їх повний перелік.

3. Web-форма «Анкети працівників» призначена для роботи із анкетними даними працівника. Оскільки анкета працівника містить порівняно багато полів, то для редагування відповідних записів призначена окрема форма. Для зручності відображення переліку працівників є можливість фільтрування записів за підрозділом та посадою.

4.Ставка заробітної плати робітника повинна піддаватися змінам. Для системи також важливо пам’ятати всі попередні значення ставки заробітної плати працівника. Тому дана частина графічного інтерфейсу складатиметься із наступних частин:

1) загальної таблиці із переліком всі працівників та поточним значенням ставки заробітної плати для кожного з них;

2) таблиці, що представлятиме історію змін ставки заробітної плати.

Таблиця із п.1.1.4 відображатиметься шляхом вибору працівника із таблиці з 1.1.

5. Фрагмент графічного інтерфейсу «Табелювання обліку робочого часу» складатиметься із 2-х фрагментів:

1) загальний перелік працівників (із можливостями фільтрації за структурним підрозділом та посадою);

2) форма деталізації записів про навантаження працівника (із можливістю фільтрації за датою запису).

6. Форма «Розрахунок заробітної плати» призначена для відображення сумарної кількості людино-годин та суми заробітної плати для кожного працівника. При цьому вона володіє можливостями фільтрування працівників за структурним підрозділом та посадою, а також надає можливості вибору діапазону дат за якими нараховується заробітна плата.

2 ТЕСТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ

Оскільки на сьогодні тенденція така, що веб-системи є важливим інструментом та мають важливе значення для ведення будь-якої діяльності, питання коректності їх тестування потребує все більшої уваги з боку зацікавлених суб'єктів [51,54]. Тестування сайтів як процес корисна та важлива для розробників процедура яка дає можливість виявляти слабкі місця проєкту, бачити шляхи вдосконалення системи, за побажаннями зробити систему більш зручною та «дружньою» до кінцевого користувача.

В цілому ж якщо розглядати роботу системи то тут є два головних аспекти, сюди можна віднести функціональність з інтерфейсом включно, за допомогою якої користувач отримує доступ до контенту та сервісів; а також доступність з доступом до сайту і швидкості завантаження сторінок включно. Для більшості випадків вони є взаємозалежними, для прикладу здійснення певних операцій є сильно залежною від часу потрібного для завантаження системи, тоді як час необхідний для завантаження системи безпосередньо може залежати від користувачів, які можуть використовувати різні функції таким чином створюючи навантаження на сервер.

В будь-якій роботі поспіх шкодить. Точно так само це працює і в IT галузі. Зрозуміло що хочеться чим швидше і деякі розробники саме так і поступають, і поспішно починають запуск відданих систем одразу після створення базового каркасу. Часто також трапляється і так, що поспіх заразом із недостатнім чи взагалі пропущеним попереднім тестуванням розроблюваного продукту може призвести до не підготовленості, розробка просто не витримує трафіка, що спричинить важкі наслідки. Взагалі багато чого в професійній діяльності залежить від Вашої історії, і якщо у вашої розробки буде низький рейтинг, а ще вона не завантажуватиметься відразу і швидко чи взагалі висне – це може негативно відбитися на розвитку вашої фірми аж до її закриття, якщо ситуація повторюватиметься.

Відомо що тестування на функціональність можна провести і в ручному режимі вже на початкових етапах розробки, але тест на доступ потребує від розробника виконання дещо складніших дій. Зазвичай основою тестування є ідея простої перевірки розробки та її параметрів при різних умовах навантаження на сервер. Щоби провести процедуру тестування на доступ, потрібно мати велику кількість відвідувачів, які використовуватимуть різноманітні браузери при огляді Інтернету, проводитимуть цей огляд при різній швидкості Інтернету, перебуваючи в різних країнах світу. Такий вид тестування можливий, проте вимагає задіяння надто великої кількості ресурсів, саме по цій причині я надала перевагу автоматичному тестуванню.

Варта зазначити, що існує безліч варіантів, коли віддалено цей процес можуть зробити і без вашої участі, проте щоб пересвідчитись у тому, що отримані результати є коректними і точними, все таки я прийняла рішення розробити свої власні сценарії тестування, а корегувати їх вже в процесі. Робитиму я це таким чином щоби сценарій тестування включав усі можливі варіанти дій, оскільки використання різного функціоналу може призвести до відкриття нових сторінок чи вкладок, а це може суттєво вплинути на серверне навантаження і роботу системи в цілому.

У деяких із сучасних утиліт тестування сайтів, в наявності є інструменти для запису поведінки, за допомогою яких можна фіксувати через запис поведінку користувачів саме під час перебування на сторінці. Така особливість дає на мою думку багато переваг однією з яких те, що весь процес для більшості випадків може здійснюватись однією людиною. Саме спосіб тестування через записування поведінки є найбільш швидким та ефективним способом для створення сценарію тестування системи. Крім цього я як розробник зможу здійснювати таку операцію, одночасно проводячи тестування особливостей розробки. Після цього варта визначитись із кількістю користувачів, які діятимуть відповідно до цього сценарію і виконати запуск програми, яка вже буде сама симулювати справжніх відвідувачів і збиратиме статистичні дані, звичайно дотримуючись певних умов та правил.

Варта також зазначити, що все програмне забезпечення призначене для тестування різних веб-ресурсів, які можуть володіти різним інтерфейсом та особливостями. Проте основним принципом їх роботи залишається створенню значного навантаження на додатки серверу таким чином виявляючи усі слабкі місця системи. Саме завдяки цьому розробнику стає очевидно над чим ще варта попрацювати. Окрім тестування поточних параметрів є утиліти за допомогою яких можна передбачити передбачати майбутню роботу, момент, коли саме система набуде популярності та зможе привертати дедалі більше кількість відвідувачів.

Таким чином стає зрозуміло, що основним призначенням процесу тестування є те, що процес відвідування їх стане легшим, зручнішим, інформативнішим та кориснішим, тому я можу говорити про те, що тестування має велике значення для процесу розробки програмного продукту. В цілому ж воно представляє собою собою поєднання людського керування та можливостей комп’ютера, яке дасть змогу мені створювати якісні і оптимізовані продукти, які обов’язково стануть популярними і визнаними веб-ресурсами.

2.1 Підхід до розробка тестів

Тестування на доступність веб вважається як підмножина тестування використовності, коли користувачі, які беруться до уваги, володіють функціональними обмеженнями, які можуть впливати на те, як вони використовують безпосередньо веб. Кінцевою метою як використовності, так і доступності, є виявлення, наскільки зручно та легко потенційні користувачі можуть використати веб-сайт, та використання цієї інформації з метою покращення можливих майбутніх як реалізацій так і конструктивних рішень.

Як правило, оцінка доступності є більше формалізованою, ніж тестування використовності. Природно що і Закон так само як і громадська думка не толерують щодо дискримінації людей із функціональними обмеженнями, особливо в наш час, коли триває війна і, на жаль, тенденція невтішна і таких людей стає все більше. Вимальовується цілий окремий напрямок, і випадки вже набувають масового характеру. Саме тому у свій роботі я приділила аналізу та врахуванню потреб таких людей, і вважаю, вони не мають відчувати ніякого дискомфорту. Таким чином, щоби нікого не обмежувати та не виділяти, урядом та іншими організаціями підтримується дотримання різноманітних стандартів щодо доступності веб.

Проте, потрібно розрізняти та розділяти дотримання стандарту та підвищення на максимум доступності до веб ресурсу. Звичайно ідеально, якщо ці два критерії співпадуть, проте може трапитись так, що будь-який даний стандарт не спрацює і цьому є ряд причин, зокрема:

1- беруться до уваги потреби людей із широким спектром недоліків.

2- робиться спроба прирівняти потреби людей із широким спектром обмежень по можливостям.

3- оптимізація, коли робиться спроба «підігнати» потреби людей із вадами під оптимальні методи.

4- з метою вираження потреб чи методів використовується обмежена мова.

Такі тонкощі, які можна вважати і недоліком, оскільки ними можуть скористуватись не добросовісні розробники для введення на ринок продуктів не відповідної якості.

Крім вище сказаного потрібно розуміти, що доступність веб ресурсу є метою, і не повинне сприйматися як налаштування типу так / ні. Власне доступність можна вважати сполучною ланкою між людськими потребами та технологіями. Саме з розвитком розуміння та вирішення людських потреб та адаптації технології під вирішення таких потреб, вимоги до доступності також можуть і будуть змінюватись, і заразом із цими змінами будуть морально

старіти і стандарти. Таким чином можу стверджувати, що веб ресурси різного призначення та різноманітні мережі забезпечують різні потреби використовуючи при цьому різні технології. Наприклад, голосові чати відмінно підійдуть для людей із проблемами із зором, тоді як відео чати реально допоможуть користувачам із вадами слуху, оскільки вони спілкуються здебільшого мовою жестів.

При виявленні певних недоліків повстає специфічне запитання з'ясування того наскільки зручно та легко використовувати представлений програмний продукт, оскільки може провокуватись якийсь додатковий розрив між сприйняттям безпосередньо користувачів та експертів, які здійснюють оцінку. При оцінці доступності варта приймати до уваги те що саме означає сприймати Web при допомозі як інших почуттів так і пізнавальних можливостей, різних не типових конфігураційних можливостей та спеціального програмного забезпечення, при допомозі якого забезпечується доступ до Web для людей із певними фізичними вадами та обмеженими можливостями.

Крім цього, якщо передбачити оцінку юзабіліті або доступності розроблюваного Web-ресурсу, з точки зору користувачів різної вікової категорії, то вона буде не порівнюваною, навіть з погляду вікових можливостей та досвіду, проте базові потреби людей, із частково обмеженими можливостями співпадатимуть навіть для різних цільових аудиторій, зокрема спеціальна технологія для зчитування екрану, яка може бути просто невідома фахівцю, який здійснює оцінку, проте, за допомогою якої здійснюється взаємодія середовища персонального комп'ютера зі браузером Web.

Тому саме рекомендації на доступність та інструменти дають змогу мінімізувати таких розрив сприйняття. Проте, вони лише доповнюють, і аж ніяк не заміняють, і тут важливі уява, технічна винахідливість і комунікація із користувачами.

Стосовно того коли саме повинна виконуватись тестування, можу відповісти зі свого досвіду: почати чим раніше, і повторювати якомога частіше.

Оскільки тестування на фінальному з етапів створення програмного продукта може призвести до того, що проєкт може затягнеться, наслідком чого буде перевитрати бюджета, а при погано спланованому тестуванні наслідки взагалі передбачити важко, але однозначно вони будуть негативними, аж до зриву цілого проєкту.

2.2 План тестування

Існує чотири основних методи тестування: тест на зручність використання (вимагає ресурсів або часу, оскільки потрібно залучити велику кількість користувачів для об’єктивної оцінки), перевірка коректності посилань, перевірка на стійкість до великих навантажень та перевірка вірності HTML коду. Це дозволить перевірити на коректність відображення ваш веб-проект у різних браузерах, та дозволить усунути неполадки, для коректного сприйняття сайту різною категорією користувачів.

Для ретельно тестування сайту потрібно його відкрити в ряді найбільш популярних браузерів. Кожний браузер по різному відображає одні і ті ж елементи через використання різних “двигунів” та опрацювання таблиць стилів.

Досвідчені Web-розробники перевіряють у своїх проєктах кожну сторінку і кожне посилання в декількох різних браузерах. Усі зв'язки всередині вашого сайту можна легко перевірити за допомогою команди перевірити гіперпосилання чи протоколу стану гіперпосилань провідника.

Тест на зручність використання має суб’єктивну долю оцінки, адже кожен користувач по-іншому оцінює зручність реалізації функцій сайту, дизайн.

2.3 Виконання тестів

Процес тестування проводився методом відкриття сторінки в браузерах різних ОС та оглядом усіх елементів сторінки на правильність відображення.

Головна сторінка сайту на iOS відображається нормально.

Головна сторінка сайту у браузері MozillaFirefox на ОС Linux відображається нормально.

Головна сторінка сайту у браузері Google Chrome на ОС Windows відображається нормально.

Під час тестування у різних браузерах на різних ОС сторінки сайту відкривалися добре.

Таким чином, очевидним висновком, який я зробила, є те, що якість не є чимось абсолютним, все таки вона є в достатній мірі суб’єктивним поняттям. Саме тому я вважаю, що тестування, якщо цей процес розглядати як процес по своєчасному виявленню як помилок так різного роду дефектів, не може в повній мірі забезпечувати коректність усього розроблюваного програмного продукту. При виконанні цього процесу лише виконується порівняння на відповідність зі специфікацією стану та поведінки програмного продукту. Крім цього варта чітко розмежовувати саму процедуру тестування програмної розробки від безпосереднього забезпечення та дотримання в процесі якості всіх етапів створення розроблюваного програмного продукту, власне до якого і належать усі складові процесу, а не лише тестування як таке.

3 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

3.1 Охорона праці

Темою кваліфікаційної роботи є “Розробка універсальної мобільної Web-системи автоматизації обліку ресурсів бізнес-структури”. Розробка проводиться з урахуванням основних етапів життєвого циклу ПЗ. При використанні ПЗ, яке є результатом даної розробки, як і при використанні будь-якого іншого ПЗ, необхідно дотримуватися вимог з охорони праці при роботі з ПК. Розглянемо основні нормативні документи, в яких зазначені вимоги до робочих місць та приміщень при використанні ПК.

Робочі місця працівників, обладнані персональними комп’ютерами, повинні відповідати вимогам «Правил охорони праці під час експлуатації електронно-обчислювальних машин», затверджених Наказом Державного комітету України з промислової безпеки, охорони праці та гірничого нагляду (Правила), та «Державних санітарних правил і норм роботи з візуальними дисплейними терміналами електронно-обчислювальних машин», затверджених постановою Головного державного санітарного лікаря України (ДСанПіН). Правила поширюються на всіх суб’єктів господарювання незалежно від форм власності, які у своїй діяльності здійснюють роботу, пов’язану з персональними комп’ютерами, у тому числі на тих, які мають робочі місця, обладнані персональними комп’ютерами і периферійними пристроями. Зазначені нормативно-правові акти встановлюють санітарно-гігієнічні вимоги до приміщення, в якому розташоване робоче місце, власне до робочого місця, освітлення, рівнів вібрації і шуму, мікроклімату в приміщенні тощо [55].

Будівлі та приміщення, де розміщені робочі місця, повинні відповідати вимогам нормативно-технічної та експлуатаційної документації виробника персональних комп’ютерів ДСанПіН та Правил. Будівлі та приміщення, де

розміщені робочі місця операторів, мають бути не нижче другого ступеня вогнестійкості. Для всіх будівель і приміщень, де знаходяться робочі місця, повинно бути визначено клас зони. Відповідне позначення повинно бути нанесено на вхідних дверях кожного приміщення. Не дозволяється розташування приміщень з робочими місцями у підвалах і цокольних поверхах. Неприпустимим є розташування приміщень категорій А і Б, а також виробництв з мокрими технологічними процесами поряд з приміщеннями, де розташовуються робочі місця, а також над ними чи під ними. При цьому площа приміщення має бути не менше 6,0 кв. м. із розрахунку на одне робоче місце, а об'єм – не менше 20,0 куб. м [55].

Віконні прорізи приміщень для роботи з персональними комп'ютерами мають бути обладнані регульованими пристроями (жалюзі, завіски, зовнішні козирки. Для внутрішнього оздоблення приміщень з персональними комп'ютерами слід використовувати дифузно-відбивні матеріали з коефіцієнтами відбиття для стелі 0,7-0,8, для стін 0,5-0,6. Покриття підлоги повинне бути матовим з коефіцієнтом відбиття 0,3-0,5. Поверхня підлоги має бути рівною, неслизькою, з антистатичними властивостями. Забороняється для оздоблення інтер'єру приміщень з персональними комп'ютерами застосовувати полімерні матеріали (деревинно-стружкові плити, шпалери, що миються, рулонні синтетичні матеріали, шаруватий паперовий пластик тощо), що виділяють у повітря шкідливі хімічні речовини [55].

У приміщеннях з джерелами шкідливих виробничих факторів робочі місця операторів мають розміщуватися в ізольованих кабінах, які обладнані повітрообміном.

Заземлені конструкції, що знаходяться в приміщеннях, де розміщені робочі місця (батареї опалення, водопровідні труби, кабелі із заземленим відкритим екраном), мають бути надійно захищені діелектричними щитками або сітками з метою недопущення потрапляння працівника під напругу. Приміщення, де розміщені робочі місця, мають бути оснащені системою автоматичної пожежної сигналізації і вогнегасниками відповідно до вимог

чинного законодавства України. Проходи до засобів пожежогасіння мають бути вільними.

У приміщеннях, в яких розташовані робочі місця, слід щоденно робити вологе прибирання. Крім того, ці приміщення мають бути оснащені аптечками першої медичної допомоги, а при них мають бути обладнані побутові приміщення для відпочинку під час роботи, кімната психологічного розвантаження [56].

При розміщенні робочих столів з персональними комп'ютерами слід дотримувати:

- відстань між бічними поверхнями персональних комп'ютерів 1,2 м.;

- відстань від тильної поверхні одного персонального комп'ютера до екрана іншого – 2,5 м.

За потреби особливої концентрації уваги під час виконання робіт суміжні робочі місця операторів необхідно відділяти одне від одного перегородками висотою 1,5 – 2м.

Конструкція робочого місця користувача персонального комп'ютера має забезпечити підтримання оптимальної робочої пози офісного працівника. Конструкція робочого столу має відповідати сучасним вимогам ергономіки і забезпечувати оптимальне розміщення на робочій поверхні використовуваного обладнання (дисплея, клавіатури, принтера) і документів. Висота робочої поверхні робочого столу має регулюватися в межах 680-800 мм, а ширина і глибина – забезпечувати можливість виконання операцій у зоні досяжності моторного поля (рекомендовані розміри: 600-1400мм, глибина – 800-1000мм).
Для забезпечення захисту і досягнення нормованих рівнів комп'ютерних випромінювань необхідно застосування приекранних фільтрів, локальних світлофільтрів (засобів індивідуального захисту очей) та інших засобів захисту, що пройшли випробування в акредитованих лабораторіях і мають щорічний гігієнічний сертифікат [56].

Приміщення для роботи з персональними комп'ютерами мають бути обладнані системами опалення, кондиціонування повітря, або

припливно-витяжною вентиляцією. У приміщеннях на робочих місцях мають забезпечуватись оптимальні значення параметрів мікроклімату: температури, відносної вологості й рухливості повітря у відповідності до СН.

Рівні позитивних і негативних іонів у повітрі мають відповідати санітарно-гігієнічним нормам СН.

Для підтримки допустимих значень мікроклімату та концентрації позитивних та негативних іонів необхідно передбачати установки або прилади зволоження та/або штучної іонізації, кондиціювання повітря.

Слід передбачити обмеження прямої близькості від джерел природного та штучного освітлення. При цьому яскравість світлих поверхонь (вікна, джерела штучного освітлення), що розташовані в полі зору повинна бути не більше ніж 200 кд/м2. Необхідно обмежувати відбиту блискість на робочих поверхнях відносно джерел природного і штучного освітлення. При цьому яскравість відблисків на екрані ВДТ має не перевищувати 40 кд/м2, а яскравість стелі в разі застосування системи відбитого освітлення – 200 кд/м2 [56].

В кваліфікаційній роботі було реалізовано розробку універсальної мобільної Web-системи автоматизації обліку ресурсів бізнес-структури. Розробка проводилась з урахуванням основних етапів життєвого циклу ПЗ, тому важливо було розглянути основні вимоги до приміщень та робочих місць, де використовують ПК, що й було зроблено в цьому підрозділі. Також були наведені правила електробезпеки під час роботи з ПК та вимоги до пожежної безпеки в приміщенні. Щоб гарантувати безпечні умови праці під час розробки програмного забезпечення, необхідно дотримуватися наведених вимог.

3.2 Безпека в надзвичайних ситуаціях

3.2.1 Визначення, причини виникнення та класифікації надзвичайних ситуацій

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

Надзвичайна ситуація (НС) – порушення нормальних умов життя і діяльності людей на об’єкті або території, спричинене аварією, катастрофою, стихійним лихом, епідемією, епізоотією, епіфітотією, великою пожежею, застосуванням засобів ураження, що призвели або можуть призвести до людських і матеріальних втрат [57].

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

Фактори, які призводять до надзвичайних ситуацій можуть бути:

- прямі – несуть загрозу для людей, навколишнього середовища та економічних об'єктів (удар, вибух тощо);
- непрямі – діють опосередковано (ожеледиця, злива), викликаючи інші небезпечні фактори. Наприклад, обледеніння, яке само по собі не несе небезпеки людині, викликало руйнування електросистеми у кількох областях України, що призвело до припинення господарської діяльності, значних матеріальних збитків, пов'язаних з відновленням ліній електропередач, не випуском продукції підприємствами, а також створило загрозу для життя та здоров'я людей через порушення теплозабезпечення будинків.

Будь-яка з надзвичайних ситуацій може стати причиною виникнення іншої та викликати небезпечні екологічні наслідки: соціальні, природні, техногенні, небезпечні екологічні наслідки [57].

Наприклад, землетрус – природна НС – призводить до руйнування споруд, пожеж, що характерно для техногенної надзвичайної ситуації, крім того, під час землетрусу гине багато людей, руйнуються житлові будинки, інфраструктура

життєзабезпечення, що викликає соціальну НС. Дані свідчать, що в сучасних умовах практично будь-яка надзвичайна ситуація є комплексною. Визначення причин та закономірностей розвитку таких надзвичайних ситуацій є складним завданням.

У кожному конкретному випадку надзвичайні ситуації виникають через ряд причин, які можна узагальнити. Природні надзвичайні ситуації в більшості є наслідком закономірного розвитку природних метеорологічних, космічних, гідрологічних чи тектонічних процесів. Це урагани, землетруси, обвали, падіння космічних тіл тощо.

Причини виникнення природних надзвичайних ситуацій та небезпечних явищ: Закономірні природні процеси, негативний антропогенний вплив на розвиток природних процесів, Випадковість у розвитку природних процесів

Але все частіше причинами природних надзвичайних ситуацій виступає людська діяльність. Техногенний розвиток досяг такого рівня, що можна штучно викликати великі природні надзвичайні ситуації будь-якого характеру, наприклад, землетруси, цунамі, засухи, епідемії тощо.

Група дослідників страхового товариства Munchener Ruck (Німеччина) понад 25 років збирала й аналізувала дані про природні катаклізми (землетруси, шторми, повені, виверження вулканів, тайфуни), що відбулися в світі з часів Римської імперії. "Немає жодних сумнівів у тому, що частота і сила руйнування природних катастроф значно зростають", - роблять висновок автори. Причини для цього, на їх думку, створює сама людина. Перш за все, йдеться про урбаністичний розвиток, використання потенційно небезпечних територій, зміни клімату та стану довкілля.

Те, що за таке дослідження взялися співробітники страхового товариства, пояснюється тим, що кожна катастрофа приносить не лише економічні збитки, а й зменшує доходи у страховому бізнесі. З кінця шістдесятих років виплати страхових компаній на покриття збитків від природних катаклізмів у всьому світі зросли від одного до більше десяти мільярдів доларів на рік.

Виникнення соціальних надзвичайних ситуацій, перш за все, пов'язують з поширенням ідей, що часто носять антисоціальний та відверто людиноненависницький характер [58]. До соціальних конфліктів також призводять національні, економічні, псевдо релігійні та політичні причини. Війна в Чечні, Афганістані, Іраку, Палестино-Ізраїльський конфлікт, військові дії в Іраку є наочною ілюстрацією цього з тією різницею, що для кожного конфлікту вагомість причин різна. В ряді випадків причинами соціальних НС, а саме, страйків, забастовок є економіка та політика.

Причини виникнення соціальних надзвичайних ситуацій: політичні, економічні, національні, релігійні ідеологічні.

Внаслідок природних та техногенних катастроф теж виникають соціальні НС через порушення у функціонуванні систем життєзабезпечення, величезних матеріальних збитків, значного травматизму населення, психологічних факторів.

Соціальною надзвичайною ситуацією, масштаби проявлення якої останнім часом збільшуються, є тероризм. Причини тероризму криються у жадобі - влади, слави, багатства, використовуючи будь-які методи. Тероризм може експлуатувати будь-яку ідею, для формування якої використовує різноманітні расистські, нацистські, спотворені релігійні, національні, економічні, політичні, соціальні погляди, в т. ч. ідеї національного визволення, соціалізму, комунізму. рівності людей, вищої раси, формування "нового порядку", відвернення планетарної катастрофи, антиглобалізму тощо.

Збільшення кількості підприємств, ускладнення технічних систем та зростання їх потужності, розширення транспортних мереж, урбанізація на тлі недостатньої уваги до питань безпеки закономірно призводять до зростання кількості техногенних надзвичайних ситуацій, аварій та катастроф. Збільшення кількості населення на планеті та його концентрація у промислово розвинених регіонах створює умови для зростання кількості жертв під час надзвичайних ситуацій техногенного характер [58].

Усі причини, через які виникають надзвичайні ситуації техногенного характеру можна розділити на три групи:

- технічні – недосконалість, застарілість конструкцій;
- природні – специфічні метеорологічні, гідрологічні чи тектонічні умови, природні надзвичайні ситуації, випадковості (наприклад, однією з причин катастроф авіалайнерів є потрапляння птахів у двигун літака);
- антропогенні ("людський фактор") – недотримання правил безпеки, помилки, необережність, халатність.

Причини виникнення техногенних надзвичайних ситуацій: недодержання правил безпеки та необережність, недосконалість у проектуванні, кримінальні елементи та тероризм, воєнні дії, природні явища.

ВИСНОВКИ

За тему моєї кваліфікаційної роботи я обрала розробку універсальної мобільної Web-системи автоматизації обліку ресурсів бізнес-структури.

Основним завданням, яке я сформулювала стосовно моєї системи є автоматизація обліку ресурсів на прикладі одного із непоновлюваних основних – обліку тривалості роботи та заробітної оплати відповідно працівників бізнес-структури. В цілому ж варта зазначити, що робота з даними є актуальною та важливою задачею, і проблеми з цим стосуються якщо не всіх, то більшості напрямків діяльності.

Розроблюваний програмний продукт представлятиме собою Web-систему, що призначена для використанням адміністраторами та працівниками бізнес-структури, наприклад, підприємства. Зокрема, адміністратори мають право для встановлення часу по кожному працівникові, кожен із яких, відповідно, повинен звітуватися за виконану упродовж дня роботу відповідно до запланованої.

До основних завдань я віднесла:

1 – коректне фактичному встановлення виробітку та затраченого на виконання видів робіт часу;

2 – збір відомостей про виконання запланованих об’ємів певних видів робіт;

3 – точне, коректне та своєчасне нарахування для кожного із працівників;

4 – систематична перевірка та контроль коректності та відповідності використання фонда заробітної плати.

До основних передумов автоматизації обраного напрямку я віднесла: врахування можливого суттєвого об’єму та видів форм як вхідної так і вихідної документації; високу трудомісткість самого процесу збору, реєстрації так само як і опрацювання інформації; багатогранність задач, що потребують розв’язку та їх значимість; значні об’єми вхідної інформації необхідної для коректного

нарахування, стягнень та утримань; ймовірна обчислювальна складність виконання таких розрахунків.

В процесі розробки я дійшла висновку про те, що об’єктно-орієнтовний аналіз та проектування є однією із найбільш ефективних та зручних методик розробки корпоративних інформаційних систем. Реалізація програмних продуктів такого класу на платформі .NET також розкриває ряд переваг, зокрема, зручність розробки програм ного коду в середовищі MS Visual Studio, а також по тужна стандартна бібліотека фундаментальних класів. Використані су часні .NET-технології, зокрема, ASP.NET та LINQ, як показала практика розробки, досить зручно застосовувати в рамках одного программного проекту.

Вдосконалення представленого на розгляд комісії програмного продукту прогнозується в напрямку розробки глобальної корпоративної системи, яка б охоплювала би й інші виробничі процеси. Однак, цьому зрозуміло, що розробка такої масштабної системи вимагатиме значних трудозатрат, і тому її модернізація це процес достатньо довгий та розрахований на роботу в команді. Для мінімізації ризиків я думаю можна планувати нарощення функціональності через частий випуск невеликих релізів системи.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. М.Р. Петрик, Д.М. Михалик, О.Ю. Петрик, Г.Б. Цуприк. Методичні вказівки до виконання атестаційної роботи магістра за спеціальністю 121 – “Інженерія програмного забезпечення” для усіх форм навчання [Текст] – Тернопіль : Тернопільський національний технічний університет імені Івана Пулюя – 2020 – 27 с.

2. Pasika DR, Tsupryk HB (2023) Rozrobka odnostorinkovoho veb-zastosunku z vykorystanniam VUE. JS i REACT: porivnialnyi analiz produktyvnostib korystuvatskoho dosvidu ta dotsilnosti [Developing a single-page web application using VUE. JS and REACT: a comparative analysis of user experience performance and expediency]. IMSTT (Tern., 13-14 December 2023), pp. 218-219 [in Ukrainian].

3. Methods of constructing algorithms for comparative test statistical verification of mathematical models of bioobject responses to low-intensity stimuli ЄБ Яворська, БІ Яворський, ГБ Цуприк, РВ Кінаш - Scientific Journal of TNTU, 2023

4. Дискретні структури (Алгебраїчні та числові системи, комбінаторний аналіз) : навчально-методичний посібник для студентів спеціальності 121 «Інженерія програмного забезпечення», аспірантів та викладачів вищих навчальних закладів / Укладач : Бойко І.В., Петрик М.Р., Цуприк Г.Б. – Тернопіль : Тернопільський національний технічний університет імені Івана Пулюя, 2017 – 64 с.

5. Моделювання та видобуток даних (висопродуктивні обчислення у великих алгебраїчних та числових системах, комбінаторному аналізі): навчальний посібник. Тернопіль: ТНТУ 2019 – 62 с.

6. Природа фірми: походження, еволюція, розвиток / За ред. О.Е. Вільямсона, С.Дж. Вінтера / Пер. з англ. А.В. Куликова; наук. ред., пер. В.П. Кузьменко. – К.: А.С.К., 2002. 336 с.

7. Доктрина інформаційної безпеки України. Указ Президента України від 25 лютого 2017 року.

8. Пушкар М.С., Пушкар М.Р. Інформаційні ресурси для бізнесу: формування та використання. Тернопіль: Карт – бланш, 2021. 217 с.

9. Плескач В.Л. Інформаційні системи і технології на підприємствах: підручник /В.Л. Плескач, Т.Г. Затонацька. —К. :Знання, 2011. —718с

10. Інформаційні технології видобутку даних (Data mining, високопродуктивні обчислення у складних системах): навчальний посібник ІВ Бойко, МР Петрик, Г Цуприк – 2020

11. Цуприк Г.Б. Розробка розумної системи обліку роботи комп'ютерної мережі підприємства в середовищі Embarcadero RAD Studio XE / Тимченко Є, Цуприк Г. /Матеріали X Науково-технічної конференції «Інформаційні моделі, системи та технології» – Тернопіль 7 грудня 2022. — Т. : ТНТУ, 2022. — С. 133.

12. Цуприк Г.Б. Розробка програмної підсистеми алгоритмічного перетворення даних з використанням різнорідних обчислювальних засобів мовою С / О.В. Михайлівський, Г.Б.Цуприк / Матеріали XI Науково-технічної конференції «Інформаційні моделі, системи та технології» – Тернопіль 13-14 грудня 2023. — Т. : ТНТУ, 2023. — С. 216.

13. Розробка системи забезпечення безпечного обміну інформацією з використаннм технологій веб програмування / Доскоч Н., Цуприк Г. / Матеріали X Науково-технічної конференції «Інформаційні моделі, системи та технології» – Тернопіль 7 грудня 2022. — Т. : ТНТУ, 2022. — С. 114.

14. [Електронний ресурс] – Режим доступу: URL: http://www.dossier-andreas.net/software_architecture/

15. [Електронний ресурс] – Режим доступу: URL: Fielding Roy. Architectural Styles and the Design of Network-based Software Architectures. — Каліфорнійський університет в Ірвайні, 2000. — 13 December

16. Shaw, Mary; DeLine, Robert; Klein, Daniel V.; Ross, Theodore L.; Young, David M.; Zelesnik, Gregory (1995-04). Abstractions for Software Architecture and Tools to Support Them (PDF). IEEE Trans. Softw. Eng. **21** (4): 314—335. doi:10.1109/32.385970. ISSN 0098-5589.

17. James Rumbaugh, Ivar Jacobson, Grady Booch (1999). The unified modeling language reference manual (англ.) . Addison Wesley Longman Inc. ISBN 0-201-30998-X.

18. [Електронний ресурс] – Режим доступу: URL: Підручник з Umbrello UML Modeller

19. [Електронний ресурс] – Режим доступу: URL: Офіційний сайт UML

20. [Електронний ресурс] – Режим доступу: URL: Журнал «Інформаційні технології», UML: історія, специфікація, бібліографія

21. McConnel S.,Tripp L. Professional Software Engineering: Fact or Fiction? // IEEE SOFTWARE. - 1999. -Nov. - Dec. - P. 13-18.

22. David L., Parnas Software Engineering Programs Are Not Computer Science Programs // IEEE SOFTWARE. - 1999. - Nov. - Dec. - P. 19-30.

23. [Електронний ресурс] – Режим доступу: URL: http:/www.swebok.org.html.

24. Jackson M. Software requirements & specifications. - Wokingham, England: Addison-Wesley, ACM Press Books, 1995. - 228 p.

25. Merriam-Webster's New Collegiate Dictionary, 10th Edition.

26. Jotterbarn D., Miller K., Rogerson S. Software Engineering Code of Ethics is Approved // Communications of the ASM. - 1999. - V. 42. - № 10. - P. 102-107.

27. Jacobson I., Griss M., Jonsson P. Software Reuse. - N.-Y.: Addison-Wesley, 1997. - 497 p.

28. [Електронний ресурс] – Режим доступу: URL: http:/www.rational.com.uml.html

29. [Електронний ресурс] – Режим доступу: URL: http:/www.rational.com.uml

30. Sutclife A. and Maiden N. The Domain Theory for Requirement Engineering //IEEE Trans, on Software Engrg. -1998. - V. 24. - № 3. - P. 174-190.

31. Schmid H. A. Creating Applications from Components: A Manufacturing Framework Design // IEEE Software, 1996. - November. - P. 67-75.

32. Keepence B., Mannioni M. Using patterns in Model Variability in Product Families // IEEE Software -1999. - № 3. - P. 10.2-108.

33. Jacobson I., Griss M., Jonsson P. Software Reuse. - N.-Y.: Addison-Wesley, 1997. - 497 p.

34. [Електронний ресурс] – Режим доступу: URL: http:/www-ksl.Stanford.edu.html

35. [Електронний ресурс] – Режим доступу: URL: Document Object Model (DOM): definition, structure and example. IONOS Digital Guide (англ.). Процитовано 27 січня 2023.

36. [Електронний ресурс] – Режим доступу: URL: DOM Standard. dom.spec.whatwg.org. Процитовано 27 січня 2023.

37. [Електронний ресурс] – Режим доступу: URL: Introduction to Model/View/ViewModel pattern for building WPF apps.

38. Silberschatz, Abraham; Sudarshan, S. (2011). Database system concepts (вид. 6). New York: McGraw-Hill. ISBN 9780073523323. OCLC 436031093.

39. [Електронний ресурс] – Режим доступу: URL: https://www.microsoft.com/ua-ua/sql-server/sql-server-downloads

40. ISO 10002:2014. Quality management. Customer satis-faction. Guidelines for complaints handling in organizations – Revises ISO 10002:2004: introduced 15.07.2014. – ISO, 2014. – 26 p.

41. ISO/IEC 25010:2011. Systems and software engineering. Systems and software Quality Requirements and Evaluation (SQuaRE). System and software quality models – Revises ISO/IEC 9126-1:2001: introduced 01.03.2011. – ISO/IEC, 2011. – 34 p.

42. ISO/IEC 25030:2007. Software engineering. Software product Quality Requirements and Evaluation (SQuaRE). Quality requirements – Introduced 01.06.2007. – ISO/IEC, 2007. – 36 p.

43. Bourque P. Guide to the software engineering body of knowledge (SWEBOK). Version 3.0 / P.Bourque, R.E.Fairley – A project of the IEEE Computer Society, 2014. – 335 p.

44. Бйорн Страуструп . Мова програмування C++. Короткий курс - 2019. - 320 с. - ISBN 978-5-907144-12-5 .

45. Бйорн Страуструп . Програмування: принципи та практика з використанням C++. - 2016. - 1328 с. - ISBN 978-5-8459-1949-6 .

46. [Електронний ресурс] – Режим доступу: URL: https://dotnet.microsoft.com/en-us/apps/aspnet

47. Hunt, Craig. TCP/IP Network Administration. - 3rd Edition. - O'Reilly Media, Inc. - ISBN 0596002971 .

48. [Електронний ресурс] – Режим доступу: URL: https://learn.microsoft.com/ru-ru/dotnet/csharp/linq/

49. [Електронний ресурс] – Режим доступу: URL: «Visual Studio .NET» та інші продукти для розробників

50. [Електронний ресурс] – Режим доступу: URL: Розділ «Microsoft Visual Studio» на сайті Майкрософт

51. [Електронний ресурс] – Режим доступу: URL: IEEE Guide to Software Engineering Body of Knowledge, SWEBOK, 2004 (англ.)

52. [Електронний ресурс] – Режим доступу: URL: Текст лекцій до курсу «Технології розробки і тестування програм» Дідковська М. В.

53. [Електронний ресурс] – Режим доступу: URL: Quality Assurance Group — Навчання, Перекваліфікація, Підвищення рівня кваліфікацій в сфері Забезпечення Якості, Контролю Якості та Тестування ПЗ

54. [Електронний ресурс] – Режим доступу: URL: Львівська Школа тестування Програмного Забезпечення

55. Дистанційний курс «Основи охорони праці» сайту дистанційного навчання ТНТУ [Електронний ресурс]. – Режим доступу: URL: http://dl.tntu.edu.ua/index.php

56. Охорона праці : Навчальний посібник [Текст] Геврик Є.О. - Ельга: Ніка-Центр. - 2003. – 279 с.

57. Стручок В.С. Методичний посібник для здобувачів освітнього ступеня «магістр» всіх спеціальностей денної та заочної (дистанційної) форм навчання

«БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ» / Тернопіль: ФОП Паляниця В. А., –156 с. Отримано з https://elartu.tntu.edu.ua/handle/lib/39196.

58. Стручок В.С. Навчальний посібник «ТЕХНОЕКОЛОГІЯ ТА ЦИВІЛЬНА БЕЗПЕКА. ЧАСТИНА «ЦИВІЛЬНА БЕЗПЕКА»» / Тернопіль: ФОП Паляниця В. А., – 156 с. Отримано з http://elartu.tntu.edu.ua/handle/lib/39424

ДОДАТКИ

Реалізація аутентифікації користувача

```
using System;
using System.Data;
using System.Configuration;
using System.Linq;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Xml.Linq;
using System.Threading;
using RR.BusinessLayer;
using System.Security.Principal;

namespace RR.Web
{
    public class SecurityContext
    {
        public static bool IsAuthentificated()
        {
            return (HttpContext.Current.Session["UserInfo"] != null);
        }

        public static UserInfo AuthentificatedUser
        {
            get {return (UserInfo)HttpContext.Current.Session["UserInfo"]; }
            private set { HttpContext.Current.Session.Add("UserInfo", value); }
        }

        public static bool IsUserInRole(UserRole role)
        {
            return (AuthentificatedUser.Role == role);
        }

        private static string GetUserID(UserInfo userInfo)
        {
            return userInfo.Role.ToString() + userInfo.ID.ToString();
        }
        public static bool Login(string userRole, string userName, string
userPassword)
        {
            UserInfo userInfo = new UserController().GetUserlInfo(userRole,
userName, userPassword);
            if (userInfo != null)
            {
                AuthentificatedUser = userInfo;
                string userId = GetUserID(userInfo);
                HttpContext.Current.User = Thread.CurrentPrincipal =
                                           new GenericPrincipal(new
GenericIdentity(userId), new string[] {userInfo.Role.ToString()});
                FormsAuthentication.SetAuthCookie(userId, false);
                string redirectUrl = null;
                switch (userInfo.Role)
                {
                    case UserRole.Teacher: redirectUrl =
"~/FirmInfo/Default.aspx"; break;
```

```
                    case UserRole.Pupil: redirectUrl =
"~/OrderContent/Default.aspx"; break;
                    case UserRole.Guest: redirectUrl =
"~/GuestContent/Default.aspx"; break;
                }
                HttpContext.Current.Response.Redirect(redirectUrl);
                return true;
            }
            return false;
        }

        public static void Logout()
        {
            AuthentificatedUser = null;
            HttpContext.Current.User = Thread.CurrentPrincipal = null;
            FormsAuthentication.SignOut();
            HttpContext.Current.Response.Redirect("~/Default.aspx");
        }

    }
}
```

SQL-скрипт для визначення структури бази даних

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[Post]') AND type in (N'U'))
BEGIN
CREATE TABLE [dbo].[Post](
      [ID] [int] IDENTITY(1,1) NOT NULL,
      [Name] [varchar](200) NOT NULL,
 CONSTRAINT [PK_Post] PRIMARY KEY CLUSTERED
(
      [ID] ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
END
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[Department]') AND type in (N'U'))
BEGIN
CREATE TABLE [dbo].[Department](
      [ID] [int] IDENTITY(1,1) NOT NULL,
      [Name] [varchar](200) NOT NULL,
 CONSTRAINT [PK_Department] PRIMARY KEY CLUSTERED
(
      [ID] ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
END
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
```

```
IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[Employee]') AND type in (N'U'))
BEGIN
CREATE TABLE [dbo].[Employee](
	[ID] [int] IDENTITY(1,1) NOT NULL,
	[Name1] [varchar](100) NOT NULL,
	[Name2] [varchar](100) NOT NULL,
	[Name3] [varchar](100) NOT NULL,
	[DepartmentID] [int] NULL,
	[PostID] [int] NULL,
	[BirDate] [datetime] NULL,
	[PassCode] [varchar](2) NULL,
	[PassSeries] [varchar](6) NULL,
	[TaxID] [varchar](10) NULL,
	[Address] [varchar](200) NULL,
	[Tels] [varchar](200) NULL,
	[Emails] [varchar](200) NULL,
	[Education] [text] NULL,
	[JobExperience] [text] NULL,
	[Family] [text] NULL,
	[AcceptanceDate] [datetime] NOT NULL,
	[Dismission] [datetime] NULL,
 CONSTRAINT [PK_Employee] PRIMARY KEY CLUSTERED
(
	[ID] ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
END
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[WorkLog]') AND type in (N'U'))
BEGIN
CREATE TABLE [dbo].[WorkLog](
	[ID] [int] IDENTITY(1,1) NOT NULL,
	[EmployeeID] [int] NOT NULL,
	[Date] [datetime] NOT NULL,
	[Hours] [numeric](2, 1) NOT NULL DEFAULT ((8)),
	[HourSalary] [numeric](10, 2) NOT NULL,
	[Report] [text] NULL,
	[Confirm] [bit] NULL DEFAULT ((0)),
 CONSTRAINT [PK_WorkLog] PRIMARY KEY CLUSTERED
(
	[ID] ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
END
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[SalaryHistory]') AND type in (N'U'))
BEGIN
CREATE TABLE [dbo].[SalaryHistory](
	[ID] [int] IDENTITY(1,1) NOT NULL,
	[EmployeeID] [int] NOT NULL,
	[StartDate] [datetime] NOT NULL,
	[EndDate] [datetime] NULL,
	[HourSalary] [numeric](10, 2) NOT NULL,
 CONSTRAINT [PK_SalaryHistory] PRIMARY KEY CLUSTERED
(
```

```
      [ID] ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
END
GO
IF NOT EXISTS (SELECT * FROM sys.foreign_keys WHERE object_id =
OBJECT_ID(N'[dbo].[FK_Employee_Department]') AND parent_object_id =
OBJECT_ID(N'[dbo].[Employee]'))
ALTER TABLE [dbo].[Employee]  WITH CHECK ADD  CONSTRAINT [FK_Employee_Department]
FOREIGN KEY([DepartmentID])
REFERENCES [dbo].[Department] ([ID])
GO
ALTER TABLE [dbo].[Employee] CHECK CONSTRAINT [FK_Employee_Department]
GO
IF NOT EXISTS (SELECT * FROM sys.foreign_keys WHERE object_id =
OBJECT_ID(N'[dbo].[FK_Employee_Post]') AND parent_object_id =
OBJECT_ID(N'[dbo].[Employee]'))
ALTER TABLE [dbo].[Employee]  WITH CHECK ADD  CONSTRAINT [FK_Employee_Post]
FOREIGN KEY([PostID])
REFERENCES [dbo].[Post] ([ID])
GO
ALTER TABLE [dbo].[Employee] CHECK CONSTRAINT [FK_Employee_Post]
GO
IF NOT EXISTS (SELECT * FROM sys.foreign_keys WHERE object_id =
OBJECT_ID(N'[dbo].[FK_WorkLog_Employee]') AND parent_object_id =
OBJECT_ID(N'[dbo].[WorkLog]'))
ALTER TABLE [dbo].[WorkLog]  WITH CHECK ADD  CONSTRAINT [FK_WorkLog_Employee]
FOREIGN KEY([EmployeeID])
REFERENCES [dbo].[Employee] ([ID])
GO
ALTER TABLE [dbo].[WorkLog] CHECK CONSTRAINT [FK_WorkLog_Employee]
GO
IF NOT EXISTS (SELECT * FROM sys.foreign_keys WHERE object_id =
OBJECT_ID(N'[dbo].[FK_SalaryHistory_Employee]') AND parent_object_id =
OBJECT_ID(N'[dbo].[SalaryHistory]'))
ALTER TABLE [dbo].[SalaryHistory]  WITH CHECK ADD  CONSTRAINT
[FK_SalaryHistory_Employee] FOREIGN KEY([EmployeeID])
REFERENCES [dbo].[Employee] ([ID])
GO
ALTER TABLE [dbo].[SalaryHistory] CHECK CONSTRAINT [FK_SalaryHistory_Employee]
```

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
імені ІВАНА ПУЛЮЯ
НАУКОВЕ ТОВАРИСТВО ім. ШЕВЧЕНКА

М А Т Е Р І А Л И

XII НАУКОВО-ТЕХНІЧНОЇ КОНФЕРЕНЦІЇ

«ІНФОРМАЦІЙНІ МОДЕЛІ, СИСТЕМИ ТА ТЕХНОЛОГІЇ»

18-19 грудня 2024 року

ТЕРНОПІЛЬ

УДК 001
М34

ПРОГРАМНИЙ КОМІТЕТ

Голова: Приймак Микола – професор кафедри комп'ютерних систем та мереж, д.т.н., професо**р.**
Співголови: Марущак Павло – проректор з наукової роботи, докт. техн. наук, професор.
Баран Ігор – канд. техн. наук, доцент, декан факультету ФІС.
Науковий секретар: Надія Крива– старший викладач.

Члени: Василь Кривень - завідувач кафедри математичних методів в інженерії д.ф.-м.н., професор; Галина Осухівська – завідувач кафедри комп'ютерних систем та мереж, к.т.н., доцент; Микола Карпінський - професор кафедри кібербезпеки, д.т.н., професор; Жанна Баб'як - завідувач кафедри української та іноземних мов, к.пед. н., доцент; Ярослав Литвиненко – професор кафедри комп'ютерних наук, д.т.н., професор; Михайло Петрик - завідувач кафедри програмної іженерії, д.ф.-м.н., професор; Наталія Загородна – завідувач кафедри кібербезпеки, к.т.н., доцент.

ОРГАНІЗАЦІЙНИЙ КОМІТЕТ

Голова: Скоренький Юрій Любомирович – канд. техн. наук, доцент кафедри фізики.
Члени: доцент кафедри комп'ютерних наук, к.т.н. В. Никитюк; доцент кафедри програмної інженерії, к.т.н. Д. Михалик; доцент кафедри кібербезпеки, к.т.н. М. Стадник; доцент кафедри комп'ютерних систем та мереж, к.т.н. Є. Тиш; ст. викладач Л. Джиджора.

М34 Матеріали XII науково-технічної конфції «Інформаційні моделі, системи та технології» Тернопільського національного технічного університету імені Івана Пулюя, (Тернопіль, 18–19 грудня 2024 р.). – Тернопіль : Тернопільський національний технічний університет імені Івана Пулюя, 2024.

Адреса оргкомітету: ТНТУ ім. І. Пулюя, м. Тернопіль, вул. Руська, 56, 46001,
тел. (0352) 52-41-33, факс (0352) 25-49-83.
E-mail: conffis2024@gmail.com

Редагування, оформлення та верстка: Надія Крива

СЕКЦІЇ КОНФЕРЕНЦІЇ, ЯКІ ПРЕДСТВЛЕНІ В ЗБІРНИКУ

- Математичне моделювання
- Інформаційні системи та технології, кібербезпека
- Комп'ютерні системи та мережі
- Програмна інженерія та моделювання складних розподілених систем
- Новітні фізико-технічні та освітні технології

В збірнику надруковано тези доповідей XII науково-технічної конференції «Інформаційні моделі, системи та технології» (Тернопіль, 18–19 грудня 2024 р.) за такими науковими напрямками: математичне моделювання; інформаційні системи та технології, кібербезпека; комп'ютерні системи та мережі; програмна інженерія та моделювання складних розподілених систем; новітні фізико-технічні та освітні технології.

Розрахований на науковців, викладачів та студентів вузів.

За зміст тез та дотримання норм академічної доброчесності відповідальність несе автор.

УДК 004.41
В. В. Деркач; Г. Б. Цуприк, канд. техн. наук, доц.
(Тернопільський національний технічний університет імені Івана Пулюя, Україна)

РОЗРОБКА УНІВЕРСАЛЬНОЇ МОБІЛЬНОЇ WEB-СИСТЕМИ АВТОМАТИЗАЦІЇ ОБЛІКУ РЕСУРСІВ БІЗНЕС-СТРУКТУРИ

UDC 004.41
V.V. Derkach; H. B. Tsupryk, PhD, Assoc.Prof.

DEVELOPMENT OF A UNIVERSAL MOBILE WEB-SYSTEM FOR AUTOMATION OF BUSINESS-STRUCTURE RESOURCES

Автоматизована інформаційна система (системи) є одним із невід'ємних атрибутів бізнес-структур всіх цивілізованих країн, в тому числі і країн Західної Європи вже впродовж. без перебільшення, кількох останніх десятиліть. Саме їх використання підвищує ефективність, а заразом із цим і прибутковість бізнесу, тим самим збільшуючи і його конкурентоспроможність. Якщо ж розглядати світовий ринок розробок програмного забезпечення в цілому то, на сьогоднішній день, з відкритих джерел можна констатувати, що успішно працюють цілий ряд компаній, рівня корпорації, які спеціалізуються на розробці такого типу програмних систем, для реалізації якої розвивається специфічний цілий комплекс технологій, який і отримав всім вже відому назву «об'єктно-орієнтовний аналіз та проєктування». Не дивлячись на складний, в плані ведення будь-якої успішно діяльності час, віримо, що вже дуже скоро війна переможно завершиться і будемо опиратись на той факт, що сучасна динаміка розвитку бізнес-структур в саме нашій країні вимагатиме, і в найкоротший час, впровадження інформаційних систем для автоматизації їх діяльності. Саме це і було базовим при виборі теми, яка є актуальною адже пов'язана саме із розробкою програмної системи подібного класу та призначення.

Важливою характеристикою сучасних автоматизованих корпоративних систем призначених для застосування саме бізнес-структурами можна назвати їхню мобільність та загальну доступність. Для забезпечення цих таких вимог досить добре підходить середовище типу Web, оскільки, зазвичай, системи такого класу володіють Web-інтерфейсом, що робить їх мобільними та дає змогу використовувати в буд-якій точці світу де є доступ до світової мережі.

Враховуючи потреби військового часу та поствоєнного національного ринку такого продукту як програмне забезпечення та визначаючи Web як всеохоплююче інформаційне середовище, актуальною є розробка програмного забезпечення саме із підтримкою Web-інтерфейсу і конкретно для вдосконалення і пришвидшення, без будь-яких втрат, через автоматизацію, діяльності бізнес-структур.

Microsoft. NET Framework є тією платформою запропоновано використати для розробки та яка вважається базовою та такою що можна об'єднати потрібні для процесу розробки технології та інструментальні засоби.

В якості основної мови програмування при реалізації представленого програмного проєкту обрано мову C# в якості простої, сучасної, об'єктно-орієнтованої і безпечної щодо типів мови програмування, наслідувану від мов C/C++.

Серйозним аргументом на користь .NET технології є можливість керувати пам'яттю, котра базується на принципі «без сміття». Головною ідеєю цього під ходу є те, що якщо об'єкт не використовується (тобто кількість посилань на нього рівна нулю), то його слід знищити. Такий підхід знімає з розробника відповідальність за звільнення зарезервованої динамічної пам'яті і, як наслідок, значно спростити безпосередньо сам процес розробки.

Варто зауважити, що хоча й C# призначена для розробки коду, що виконується середовищем .NET, сама вона не є частиною .NET. Існують деякі можливості, які підтримуються .NET, але не підтримуються C#. І, як не дивно, існують можливості C#, які не підтримуються .NET.

В представленому програмному проєкті в якості механізму обробки даних та доступом до бази данних обрано LINQ (зокрема, LINQ to SQL). Такий вибір зумовлено тим, що ця, порівняно молода технологія, надає ряд гнучких та зрозумілих механізмів роботи з даними.

В процесі розвитку платформи .NET і підтримуваних нею мов C# та VB, стало зрозуміло, що однією із найбільше проблемних областей для розробників залишається доступ до даних із різноманітних джерел. Маніпуляція з XML та безпосередній доступ до бази данних часто, в кращому випадку, заплутані, а в гіршому взагалі проблемні.

Література

1. Плескач В.Л. Інформаційні системи і технології на підприємствах: підручник /В.Л. Плескач, Т.Г. Затонацька. —К. :Знання, 2011. —718с
3. Бойко І.В., Петрик М.Р., Цуприк Г.Б. Інформаційні технології видобутку даних (Data mining, високопродуктивні обчислення у складних системах): навчальний посібник. Тернопіль: : ТНТУ 2020 – 62 с.
4. Петрик М.Р., Михалик Д.М., Петрик О.Ю., Цуприк Г.Б. Методичні вказівки до виконання атестаційної роботи магістра за спеціальністю 121 – «Інженерія програмного забезпечення» для усіх форм навчання – Тернопіль: Тернопільський національний технічний університет імені Івана Пулюя – 2020 – 27 с.
5. Бойко І.В., Петрик М.Р., Цуприк Г.Б. Моделювання та видобуток даних (високопродуктивні обчислення у великих та числових системах, комбінаторному аналізі): навчальний посібник. Тернопіль : ТНТУ 2019 – 62 с.