UNICHECK by Turnitin

НУВГП

Ім'я користувача:
Андрій Олександрович Коломис

ID перевірки:
1015667199

Дата перевірки:
21.06.2023 13:41:17 EEST

Тип перевірки:
Doc vs Library

Дата звіту:
23.06.2023 18:31:17 EEST

ID користувача:
12558

Назва документа: **Дипломна_Коломис_ІСТ21інт.docx**

Кількість сторінок: **59** **Кількість слів:** **8442** **Кількість символів:** **67972** **Розмір файлу:** **5.81 MB** **ID файлу:** **1015312563**

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

5.29%
Схожість

Найбільша схожість: **1.87%** **з джерелом з Бібліотеки** (**ID файлу:** **1004978431**)

Пошук збігів з Інтернетом не проводився

5.29% Джерела з Бібліотеки **350** ..... **Сторінка** 61

1.47% Цитат

Цитати **7** ..... **Сторінка** 62

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Підозріле форматування **20 сторінок**

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

Допущено до захисту:
Завідувач кафедри
_____ д. е. н., проф. П. М. Грицюк
«_____» _____ 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА
на здобуття ступеня «бакалавр»
за освітньо-професійною програмою «Інформаційні системи і технології»
спеціальності 126 «Інформаційні системи та технології»

на тему: «**Розробка інформаційної системи для обліку отримання допомоги тимчасово переміщених осіб**»

Виконав:
здобувач вищої освіти 2 курсу із скороченим терміном навчання, групи ІСТ-21інт
Коломис Андрій Олександрович

Керівник:
канд. техн. наук, доцент Барановський С. В.

Рецензент:
канд. техн. наук, доцент Гладка О. М.

Рівне – 2023

Зміст

ВСТУП

Протягом воєнного періоду більшість українців отримала реальну допомогу у різних формах. У ці складні часи, небайдужі люди стають нашою опорою, коли нам це надзвичайно необхідно. Допомога надходить з різних джерел - країн, організацій та індивідуальних осіб. Проте, незважаючи на те, що часто нам допомагають без умов, так не може бути постійно, тому важливо вести облік отриманої підтримки.

Усі волонтерські центри стикаються з великим потоком ресурсів, і можливо на початку війни ці ресурси старалися використати якомога швидше, але з часом виникає необхідність у веденні обліку, плануванні та звітності про досягнуті результати. Крім того, варто зазначити, що держава регулює отримання різноманітної допомоги. Тому нам необхідно вести облік наших ресурсів.

Враховуючи це, має велике значення розібратися у різних формах отриманої допомоги, як вона є використаною і, крім того, встановити контроль за використанням ресурсів.

Існують різні терміни на рівні законодавства, тому “Благодійна допомога” та “Гуманітарна допомога” відрізняються. Розрізнення між благодійністю та гуманітарною допомогою починається на законодавчому рівні, оскільки кожна форма допомоги має власну сутність. Загальні принципи благодійництва регулюються Цивільним кодексом України та Законом України "Про благодійництво", а процедура надання, обробки, розподілу та контролю за цільовим використанням гуманітарної допомоги визначена Законом України "Про гуманітарну допомогу". Крім внутрішніх нормативних актів, в останні часи, в умовах воєнного стану, необхідно посилатися на міжнародні договори, на які Верховна Рада України дала свою згоду і які є обов'язковими для виконання.

Однак, не дивлячись на це, важливо відзначити, що гуманітарна допомога належить до сфери благодійності і має особливі характеристики та процедури

отримання. Варто зауважити, що правильним є вживання терміну "благодійність", а не "благодійна допомога"!

Закон про благодійну діяльність не надає конкретного визначення терміну "благодійна допомога", але встановлює загальне визначення благодійної діяльності. Закон "Про гуманітарну допомогу", зі свого боку, докладно пояснює, що саме вважається гуманітарною допомогою. Тоді, в чому ж полягає різниця?

«*Благодійна діяльність — добровільна особиста та/або майнова допомога для досягнення визначених цим Законом цілей, що не передбачає одержання благодійником прибутку, а також сплати будь-якої винагороди або компенсації благодійнику від імені або за дорученням бенефіціара*».[1]

«*Гуманітарна допомога — цільова адресна безоплатна допомога в грошовій або натуральній формі, у вигляді безповоротної фінансової допомоги або добровільних пожертвувань, або допомога у вигляді виконання робіт, надання послуг, що надається іноземними та вітчизняними донорами із гуманних мотивів отримувачам гуманітарної допомоги в Україні або за кордоном, які потребують її у зв'язку з соціальною незахищеністю, матеріальною незабезпеченістю, важким фінансовим становищем, виникненням надзвичайного стану, зокрема внаслідок стихійного лиха, аварій, епідемій і епізоотій, екологічних, техногенних та інших катастроф, які створюють загрозу для життя і здоров'я населення, або тяжкою хворобою конкретних фізичних осіб, а також для підготовки до збройного захисту держави та її захисту у разі збройної агресії або збройного конфлікту. Гуманітарна допомога є різновидом благодійництва і має спрямовуватися відповідно до обставин, об'єктивних потреб, згоди її отримувачів та за умови дотримання вимог статті 3 Закону України "Про благодійну діяльність та благодійні організації"*».[2]

У багатьох містах існують благодійні центри, які надають гуманітарну допомогу внутрішньо переміщеним особам. Оскільки ресурси є обмеженими, проблема обліку використання цих ресурсів стає дедалі актуальнішою, особливо в умовах зростання кількості внутрішньо переміщених осіб при

одночасному зменшенні зовнішнього потоку допомоги.

Іншою проблемою є недобросовісне використання ресурсів, коли деякі особи, які не є тими, хто потребує, незаконно нагромаджують її на свою користь, замість того, щоб передати ці ресурси тим, хто справді має потребу. Також можуть виникати ситуації, коли діяльність різних волонтерських центрів несинхронізована, що створює можливість для людей обходити містом по різних волонтерських центрах з метою надмірного збирання гуманітарної допомоги або для особистої вигоди.

Метою роботи є аналіз існуючої проблеми обліку допомоги та розробка інформаційної системи для обліку розподілу гуманітарної допомоги. Тому розглянемо декілька способів вирішення цієї проблеми, та створимо концепт об'єднання різних волонтерських центрів у мережу та встановлення спільної інформаційної системи, яка дозволить вести облік отримання різних видів гуманітарної допомоги.

Отже, завданням даної роботи є:

1. Характеристика та аналіз проблеми організації надання допомоги внутрішньо-переміщеним особам громадськими організаціями.
2. Проектування інформаційної системи для обліку отримання та роздачі гуманітарної допомоги та методи їх реалізації.
3. Розробка системи, зручної у використанні та адаптивної до потреб волонтерського центру.

Об’єктом дослідження є процес діяльності волонтерських центрів в гуманітарних цілях. **Предметом дослідження** – запровадження концепту та розробка централізованої системи обліку видачі гуманітарної допомоги.

Для успішної реалізації цієї системи, нам необхідні навички розробки програмного забезпечення для організації та обробки даних, а також створення інтерфейсів, що забезпечать функціональність інформаційної системи. Крім того, це дозволить нам здобути практичні навички роботи з популярним програмним забезпеченням для баз даних, системами управління даними та іншими супутніми технологіями. Збираючи інформацію з електронних джерел,

є можливість отримати більше знань про предмет і об'єкт дослідження. Це дає нам змогу зрозуміти, як провідні ІТ-компанії використовують нові технології. В результаті набувається більше досвіду у пошуку і організації інформації, що допоможе створити продукт, який буде корисним і підтримуваним.

У першій частині нашого проекту буде детально розглянуто дану тематику, проведено аналіз проблеми і розглянуто які існують спроби вирішити ці проблеми. У другій частині буде визначиено вимоги до нашого продукту і сформуємо концепцію його подальшого розвитку. Встановимо необхідну базу даних, яка послужить основою для подальшої розробки наших власних систем. Також, у другому розділі, буде розроблено структуру бази даних для нашої інформаційної системи та опишемо процес розробки самої бази даних. У п’ятому розділі буде детально розглянуто розробку інтерфейсів для управління базами даних у нашій системі, а також інтерфейсів для представлення даних користувачам у глобальній мережі Інтернет.

1. ХАРАКТЕРИСТИКА ТА АНАЛІЗ ПРОБЛЕМИ ОРГАНІЗАЦІЇ НАДАННЯ ДОПОМОГИ ВНУТРІШНЬО-ПЕРЕМІЩЕНИМ ОСОБАМ ГРОМАДСЬКИМИ ОРГАНІЗАЦІЯМИ.

1.1 Проблеми організації діяльності громадських організацій

Надання допомоги волонтерською організацією може стикатися з численними проблемами. Від браку ресурсів, таких як фінансові обмеження та обмежений доступ до матеріальних ресурсів, до складнощів у рекрутингу та утриманні волонтерів, координації та керування великою кількістю волонтерів та врахування потреб та очікувань бенефіціарів, волонтерські організації постійно стикаються з викликами.

Брак ресурсів може ускладнювати здатність організації надавати достатню допомогу та розширювати свою діяльність. Фінансові обмеження та нестача матеріальних ресурсів можуть обмежити їхню здатність задовольняти потреби та розподіляти ресурси ефективно.

Залучення та утримання волонтерів також може бути викликом. Залучення нових волонтерів вимагає активного просування та встановлення зв'язків з потенційними учасниками. Збереження волонтерів на довгостроковій основі потребує підтримки, створення стимулів та можливостей розвитку.

Координація та керування роботою великої кількості волонтерів також може бути складною задачею. Ефективна координація, розподіл обов'язків та забезпечення взаємодії вимагають значних зусиль та ресурсів. Крім того, врахування різноманітних потреб та очікувань бенефіціарів є важливим аспектом, який вимагає ретельного аналізу та підходу, що враховує культурні, соціальні та інші специфічні фактори.

Боротьба з проблемами обліку допомоги волонтерською організацією може мати багато методів. Самий популярний і старий метод певно що буде в використанням паперового документообігу. В будь якому випадку це може бути викликом, але можна піти наступним шляхом для полегшення ситуації:

1) Оптимізація процесів: Важливо аналізувати та оптимізувати кожен крок документообігу, зосереджуючись на мінімізації зайвих процедур та заповненні форм. Ідентифікувати найбільш часо- та ресурсомісткі етапи та шукайте способи їх спрощення.
2) Упорядкування та зберігання документів: Ретельно структуризація та організація документів, встановлюючи систему каталогів та файлів, щоб легко знаходити необхідну інформацію. Використовуйте маркери, теги або інші методи позначення для швидкого доступу до важливих документів.
3) Забезпечення консистентності: Встановіть чіткі правила та процедури щодо стандартизації документації. Це допоможе забезпечити, що всі необхідні дані вказані на документах, а також зменшить можливість помилок та неправильної інтерпретації.
4) Регулярні огляди та аудити: Проводьте періодичні огляди та аудити документообігу, щоб виявити можливі проблеми та зробити вдосконалення. Визначайте слабкі місця та шукайте способи їх вирішення.
5) Підвищення свідомості та навчання: Забезпечуйте навчання та підтримку персоналу стосовно ефективного управління документацією. Переконайтесь, що всі співробітники розуміють процедури та правила документообігу.

З однієї сторони може вважатися що це самий дешевий підхід. Можливо на початку у фінансовому плані так і буде, проте надалі це буде коштувати наступних проблем, які все ж призведуть до фінансових витрат:

1) Великий обсяг паперової документації: Волонтерські організації, особливо ті, які займаються значними масштабними проектами або мають велику кількість волонтерів та активностей, можуть мати значну кількість документів, що потребують обробки та зберігання. Це може призвести до перенавантаження робочих місць, просторових обмежень та складнощів у пошуку та доступі до необхідної інформації.

2) Втрата, пошкодження або зникнення документів: Паперові документи піддаються ризику втрати, пошкодження або крадіжки. Якщо важлива інформація на документах втрачається або стає недоступною, це може вплинути на здатність організації виконувати свої завдання та надавати допомогу.
3) Обмежений доступ до інформації: Паперова документація, яка зберігається на фізичних носіях, може бути доступна лише у конкретному місці або обмеженому колу осіб. Це може ускладнити швидкий та зручний доступ до інформації для працівників, волонтерів та інших зацікавлених сторін.
4) Тривалість процесів: Паперовий документообіг може бути пов'язаний зі значними затратами часу, оскільки потребує фізичного перенесення та обробки документів. Це може призвести до затримок у рішення важливих питань та виконання завдань. Довгий час обробки документів може призвести до затримки у відповіді на запити волонтерів, бенефіціарів або інших зацікавлених сторін.
5) Високі витрати на друк та зберігання, підтримка прінтерів, тощо: Паперовий документообіг вимагає значних витрат на друк, копіювання та зберігання фізичних копій документів. Витрати на папір, принтери, факси, архівні простори та інші матеріали можуть становити значну частку бюджету організації.
6) Обмежена мобільність та гнучкість: Паперовий документообіг обмежує можливості роботи волонтерів та працівників віддалено або поза офісом. Залежність від фізичних документів може ускладнювати співпрацю, коли потрібна швидка комунікація та обмін інформацією.
7) Ризик помилок та недосконалостей: Паперовий документообіг підвищує ризик помилок, таких як втрата, неправильне заповнення або недосконаліть документів. Це може призвести до недостовірної інформації, незрозумілих інструкцій та інших проблем, які можуть вплинути на якість наданої допомоги.

11

Загалом, паперовий документообіг може створювати значні проблеми для будь якої організації, ускладнюючи ефективну комунікацію, доступ до інформації та швидкість виконання завдань. Застосування електронного документообігу може бути одним із шляхів вирішення цих проблем, спрощуючи процеси, підвищуючи ефективність та полегшуючи доступ до інформації.

1.2. Проблема інформаційного забезпечення та обліку надання допомоги ВПО та постраждалим під час війни в громадських організаціях.

Нездійснення обліку видачі гуманітарної допомоги волонтерським центром може призвести до ряду серйозних проблем, які впливатимуть на ефективність та функціонування організації. Ось кілька додаткових потенційних проблем, які можуть виникнути:

1) Недоцільне використання ресурсів: Відсутність обліку може призвести до неправильного розподілу гуманітарної допомоги. Це означає, що деякі отримувачі можуть отримувати більше допомоги, ніж їм фактично потрібно, тоді як інші можуть бути позбавлені необхідної підтримки. Це може призвести до нерівності та негласної конкуренції серед отримувачів.
2) Дублювання видачі: Без обліку може виникнути ситуація, коли одні й ті самі люди отримують гуманітарну допомогу з різних джерел або в різних волонтерських центрах. Це може призвести до надмірного використання ресурсів та втрати можливості допомогти новим отримувачам.
3) Втрата даних та неточність інформації: Паперовий облік підлягає ризику втрати або пошкодження документів, що містять важливу інформацію про видачу допомоги. Це може призвести до неповної або неточної інформації про отримувачів, їх потреби та видану допомогу. Це може ускладнити подальший аналіз, планування та звітність.

4) Неспроможність виявити тривалі потреби: Без систематичного обліку може бути важко визначити тривалі потреби отримувачів гуманітарної допомоги. Не вести облік унеможливлює аналіз тенденцій та оцінку змін потреб на часі, що ускладнює планування та адаптацію програм допомоги.
5) Недостатня звітність та відповідальність: Відсутність обліку може призвести до нечіткості та недостовірності в звітах про видачу гуманітарної допомоги. Це може порушити довіру з боку донорів, урядових органів та інших зацікавлених сторін, а також ускладнити відстеження використання ресурсів та результативності програм.

Для зменшення цих проблем та забезпечення ефективного використання гуманітарної допомоги, необхідно вести точний та систематичний облік. Це можна зробити шляхом впровадження електронної системи обліку, яка забезпечить автоматизацію процесів, зменшить ризик помилок та забезпечить точність інформації. Крім того, варто забезпечити навчання та підтримку волонтерів щодо використання цієї системи, щоб забезпечити його успішне впровадження та використання у повсякденній роботі волонтерського центру.

1.3 Огляд існуючих інструментів автоматизованого обліку діяльності громадських організацій.

1С: Бухгалтерія є універсальною бухгалтерською програмою, яка надає комплексний та аналітичний облік в різних сферах діяльності. Програма дозволяє проводити аналіз об'єктів (субрахунків) обліку за їх суттю та вартістю. Вона надає можливість вводити проводки як вручну, так і автоматично, які фіксуються в журналі операцій.

Завдяки журналу транзакцій, ви можете переглядати, фільтрувати та групувати проводки за різними параметрами, обмежуючи їх часовим інтервалом. Крім того, програма підтримує різні довідково-інформаційні переліки, такі як план рахунків, перелік видів аналітичного обліку об'єктів та

аналітичних рахунків. Програма також надає можливість розрахувати підсумки проводок за різними періодами часу, включаючи квартал, рік, місяць та інші.

1С є потужною та універсальною програмною платформою, яка може бути використана для ефективного обліку матеріальних ресурсів у волонтерському центрі. Впровадження системи обліку на базі 1С допоможе вирішити багато потенційних проблем, пов'язаних з недостатнім контролем та неефективним використанням ресурсів.

Перш за все, 1С забезпечує автоматизацію багатьох рутинних процесів, пов'язаних з обліком матеріальних ресурсів. Це означає, що багато ручних операцій можуть бути замінені автоматичними процедурами, що спрощує та прискорює роботу волонтерського центру.

Крім того, 1С дозволяє централізовано керувати матеріальними ресурсами. Завдяки цьому, всі дані про ресурси, запаси, постачальників та інші важливі параметри можуть бути зосереджені в єдиній системі. Це сприяє збільшенню точності та надійності інформації, а також полегшує контроль та планування ресурсів.

1С також надає можливість створювати різноманітні звіти та аналітичні дані про обіг матеріальних ресурсів. Це дозволяє проводити детальний аналіз, виявляти тенденції та розробляти стратегії для оптимізації використання ресурсів. Такий аналіз стає важливим інструментом для прийняття обґрунтованих управлінських рішень.

Однією з переваг 1С є його гнучкість та налаштування. Систему обліку можна адаптувати до конкретних потреб волонтерського центру, включаючи розширення функціональності, налаштування робочого процесу та інтеграцію з іншими інформаційними системами.

Враховуючи всі ці переваги, використання 1С для обліку матеріальних ресурсів волонтерського центру може допомогти вести облік, забезпечити контроль та оптимізацію ресурсів, а також допомогти в прийнятті обґрунтованих управлінських рішень.

Проте використання 1С для волонтерського центру може зіткнутися з

деякими труднощами, які варто враховувати:

1. Складність впровадження: Встановлення та налаштування системи 1С може бути складним завданням, особливо для організацій, які не мають достатнього досвіду або ресурсів для проведення процесу впровадження. Необхідно мати кваліфікованих спеціалістів, які зможуть належним чином встановити та налаштувати систему відповідно до потреб волонтерського центру.
2. Вартість: Використання системи 1С може бути пов'язане зі значними витратами на ліцензії, налаштування та підтримку. Для волонтерських центрів з обмеженим бюджетом це може стати значним обмеженням у використанні системи.
3. Навчання персоналу: Впровадження 1С вимагає підготовки та навчання персоналу, який буде користуватися системою. Це може зайняти час та вимагати додаткових зусиль для забезпечення відповідного рівня компетентності в користуванні програмою.
4. Потреба в технічній підтримці: Використання 1С може вимагати наявності технічної підтримки для вирішення потенційних проблем, оновлення програмного забезпечення та забезпечення безперебійної роботи системи. Волонтерські центри можуть зіткнутися з обмеженими ресурсами для такої підтримки, що може створити труднощі.
5. Непридатність для деяких специфічних потреб: 1С є універсальною системою, але може не відповідати всім специфічним потребам волонтерського центру. Деякі особливі функціональності можуть бути відсутніми або потребувати додаткової настройки.

Усі ці труднощі варто враховувати та здійснювати обгрунтоване розглядання переваг та недоліків перед впровадженням системи 1С у волонтерський центр.

1.4. Висновки з аналізу

В нашому випадку нам потрібно визнати – дана система була розроблена в першу чергу з іншими цілями. Якщо спробувати уявити обсяг роботи для того, щоб використати дану систему в наших цілях для обліку отримання та роздачі гуманітарної допомоги, дуже швидко буде видно явно виражені недоліки, що значно вплине на швидкість та продуктивність видачі гуманітарної допомоги.

Для прикладу, можемо зустрітися з такими недоліками даної системи в нашому випадку[6]:

1. Не зручний інтерфейс для швидкого введення великої кількості інформації про людину, як отримує допомогу.
2. В багатьох аспектах даний продукт є не допрацьованим та містить помилки
3. В «1С: Бухгалтерії» ускладнений пошук помилок, зроблених під час обробки документів.
4. Програма «1С: Бухгалтерія» досить складна в освоєнні і вимагає спеціального навчання користувачів.

В нашому випадку, проживаємо в час, коли не можемо спланувати перебіг війни. Нам потрібно створити гнучку інформаційну систему, яка зможе переходити від невеликого проекту до більшого. Це пояснюється тим, що змінити робочий процес проекту важко, коли проект уже в розпалі та швидко розвивається. Легше вибрати правильну інформаційну систему на початку проекту; цю ж систему можна легко розширити для задоволення будь-яких майбутніх потреб.

2. ІНФОРМАЦІЙНІ МОДЕЛІ ТА МЕТОДИ ЇХ РЕАЛІЗАЦІЇ

2.1 Загальні вимоги щодо вибору методів розробки IT-проекту

Розробка інформаційних моделей та вибір методів їх реалізації є ключовим етапом перед початком будь-якого проекту. Це важливий процес, який допомагає забезпечити успішну реалізацію проекту та досягнення його цілей. Нижче представлені основні причини, чому варто зосередитися на розробці інформаційних моделей та виборі відповідних методів:

1. Розуміння потреб та вимог проекту: Розробка інформаційних моделей допомагає визначити та зрозуміти потреби, вимоги та цілі проекту. Вона дозволяє зробити глибокий аналіз проекту, ідентифікувати ключові елементи та встановити зв'язки між ними. Це допомагає зрозуміти, які дані потрібно збирати, як їх обробляти та які функції повинні бути включені в інформаційну систему.
2. Ефективне планування та керування проектом: Інформаційні моделі допомагають визначити логіку та послідовність виконання завдань, встановити залежності та ресурси, необхідні для успішного завершення проекту. Вони дозволяють розподілити роботу між командою, планувати терміни виконання та контролювати хід робіт.
3. Забезпечення якості та ефективності проекту: Інформаційні моделі дозволяють зрозуміти, які процеси можуть бути оптимізовані та автоматизовані, що допомагає покращити якість та ефективність виконання проекту. Вони дозволяють ідентифікувати можливі проблеми, ризики та виробляти стратегії їх запобігання.
4. Забезпечення спілкування та співпраці: Розробка інформаційних моделей допомагає створити спільну мову та зрозуміти усім учасникам проекту його основні концепції, процеси та взаємозв'язки. Це сприяє покращенню спілкування між командою та зацікавленими сторонами, а також забезпечує більш ефективну співпрацю.

5. Мінімізація помилок та ризиків: Інформаційні моделі дозволяють зрозуміти, які процеси можуть бути джерелом помилок та ризиків, і допомагають виявити та усунути їх ще до початку реалізації проекту. Вони дозволяють провести аналіз можливих проблем та забезпечити введення необхідних заходів для їх запобігання.
6. Забезпечення масштабованості та майбутнього розвитку: Розробка інформаційних моделей допомагає побудувати систему, яка буде легко масштабуватися та адаптуватися до змінних потреб проекту. Вона дозволяє передбачити майбутні зміни та врахувати їх при проектуванні системи.

Узагальнюючи, розробка інформаційних моделей та вибір методів їх реалізації перед початком проекту є важливим кроком для забезпечення успішної реалізації проекту, ефективного планування та керування, покращення якості та ефективності, забезпечення спілкування та співпраці, мінімізації помилок та ризиків, а також масштабованості та майбутнього розвитку проекту.

2.2 Специфікація вимог до програмного продукту.

Специфікація вимог — це акт документування всіх вимог до системи та користувача в документі. Вони мають бути чіткими, послідовними, повними та вичерпними, щоб забезпечити точний кінцевий продукт.

Після збору всіх необхідних вимог, вимоги мають бути проаналізовані та проговорені. Має бути створений офіційний документ, який пояснює ці вимоги після того, як проект завершений. Специфікація вимог — це процес документування потреб і обмежень системи або користувача в точних термінах.

Існують такі види специфікацій: функціональні

Документ із функціональними вимогами визначає, як нова система буде виглядати та працювати на функціональному рівні. Він описує, на які команди реагує система, чого від неї очікують користувачі та як вона функціонує. Цей опис допомагає створити чітке уявлення про те, як користувачі хочуть, щоб

система функціонувала.

Нефункціональні вимоги описують обмеження системи, що створюється. Ці вимоги не впливають на функціональність програми. Також, існує загально-відомий список класифікації нефункціональних вимог на різні категорії, наприклад:

1. Інтерфейс користувача
2. Надійність
3. Безпека
4. Продуктивність
5. Технічне обслуговування
6. Стандарти

2.3 Функціональні вимоги обліку

Функціональний вимоги деталізують функції, які має виконувати система. Ці критерії не визначають, як, коли або що створювати — вони визначають, «що буде створено».

Функціональні вимоги починаються з твердження, яке вказує на важливість функціональності для програми. Ідеальним є створення робочого прототипу, оскільки це допоможе визначити, скільки роботи ще залишилося. Після цього має початися розробка. Функціональна вимога не зосереджується на технічних деталях; натомість він зосереджується на призначених для кінцевих користувачів функціях. Ці зміни потенційно можуть відбутися під час проекту, тому функціональні вимоги не включають внутрішньої логіки.[4]

В нашому випадку, система обліку допомоги повинна:

1. Надавати можливість реєструвати людей. Кожна людина має таку обов’язкову інформацію про себе:
    a. Ім’я
    b. Прізвище
    c. Поштова скринька

d. Нікнейм

2. Користувач може бути тимчасово зареєстрованим в різних населених пунктах, тому інформація доповнюється окремою таблицею:
    a. Хто реєструється
    b. Сервісний центр реєстрації
    c. Дата реєстрації
3. Дати можливість додавати місця реєстрації
    a. Ім’я населеного пункту
    b. Розмір
    c. Статус небезпеки
4. Можливість зареєструвати волонтерський центр
    a. Місце реєстрації
    b. Ім’я волонтерського центру
    c. Адреса волонтерського центру
5. Надання статусів для волонтерських центрів враховуючи їхні поточні можливості:
    a. Волонтерський центр
    b. Категорія допомоги
    c. Опис допомоги
    d. Відповідальна людина
6. Фіксація факту надання допомоги
    a. Особа, якій була надана допомога
    b. Вид допомоги, який було надано
    c. Волонтерський центр, в якому було надано допомогу
    d. Обсяг наданої допомоги
    e. Дата

Маючи даний перелік даних, є змога знайти також інші дані, для прикладу:

1. Періодичність візитів тієї чи іншої людини у волонтерський центр.
2. Тенденцію зміни потреб кожної людини

3. Продуктивність того чи іншого волонтерських центрів в різних регіонах країни
4. Статистику актуальності тієї чи іншої категорії потреб по регіонах, де є наявні волонтерські центри.
5. Можливе перенасичення кількість ВПО на один регіон, що дасть змогу передбачити наплив людей і дасть можливість зреагувати наперед, забезпечивши ресурсами з інших волонтерських центрів.
6. Та ін.

2.4 Нефункціональні вимоги обліку

1. Мінімалістичний та зрозумілий дизайн.
2. Мінімальна кількість інформації на ввід
3. Сумісна з більшістю популярних браузерів, для прикладу, Safari, Microsoft Edge, Google Chrome, Firefox
4. Є можливість використання на мобільних девайсах

2.5 Моделювання предметної області на логічному рівні

Бази даних повинні відповідати своєму призначенню, точно відображаючи дизайн. Процес проектування вимагає двох етапів: фізичного та логічного моделювання. Фізичний етап моделює основний предмет у зв'язку з логікою. Далі йде логічне моделювання, яке формує структуру, що відповідає вимогам користувача.

Наша цільова інформаційна система спирається на ER-діаграми[3]. Ці діаграми створюють зображення, що представляє сутності нашої бази даних та їхні зв'язки один з одним. Це можна побачити на рис. 1 ER-діаграма, яка була створена, відображає сутності нашої бази даних. Кожна з цих діаграм необхідна для функціонування системи та досягнення наших цілей

21

Рис. 2.5.1 (ER-діаграма сутності зв’язків бази даних)

Ми розглянемо кожен інформаційний об’єкт окремо та обговоримо його роль в інформаційній системі та зв’язок з іншими об’єктами.

People

Об’єкт особи, person – це сутність, що описує особу, який зареєструвана у нашій системі. Дана сутність використовується у випадку опису тимчасово переміщеної особи та також відповідальної особи за той чи інший напрямок допомоги у волонтерському центрі. Цей об’єкт містить такі поля:

1. PersonId
2. Name
3. LastName
4. Email
5. UserName
6. Password
7. CreatedAt
8. UpdatedAt

PersonId –унікальне поле, що буде генеруватися базою даних для цілісності та унікальності та містить унікальний ідентифікатор користувача. Використовується як primary key.

Name - поле, що містить ім'я особи.

LastName – поле, що містить прізвище особи.

Email – поле, що містить електронну пошту особи.

UserName – поле, що містить довільний «нікнейм» особи.

Password – поле, в якому зберігається хешована версія паролю користувача.

CreatedAt – дата першої реєстрації особи.

UpdatedAt – дата оновлення даних про особу.

PeopleRegistrations

PeopleRegistrations – це проміжна сутність, що використовується для зв'язки таблиці **People** та **ServiceCenter** з відношенням багато до багатьох.
Цей об'єкт містить наступні поля:

1. RegistrationId
2. UserId
3. ServiceCenterId
4. RegistrationId

RegistrationId – 16-ти бітне унікальне поле, що буде генеруватися базою даних для цілісності та унікальності та містить унікальний ідентифікатор запису в цій таблиці. Використовується як primary key.

PersonId – 16-ти бітне унікальне поле, що буде генеруватися базою даних для цілісності та унікальності та містить унікальний ідентифікатор користувача. Є ключем foreign key з посиланням на таблицю **People.**

ServiceCenterId – 16-ти бітне унікальне поле, що буде генеруватися базою даних для цілісності та унікальності та містить унікальний ідентифікатор сервісного центру. Є ключем Foreign key з посиланням на таблицю **ServiceCenter**.

RegistrationDate – дата реєстрації особи в конкретному сервісному центрі. Відбувається при тимчасовому вимушеному переїзді.

ServiceCenter

ServiceCenter – це таблиця, що містить короткий опис сервісного центру

Дана таблиця буде містити наступні поля:

1. ServiceCenterId
2. RegionId
3. SattlementName

ServiceCenterId – унікальний ідентифікатор сервісного центру, де будуть реєструватися особи. Є ключем PrimaryKey.

RegionId – унікальний ідентифікатор області. Є foreingKey з посиланням на таблицю **Regions.**

SattlementName - назва сервісного центру.

Regions

Regions – це таблиця, що містить інформацію про області країни та стан небезпеки в кожній з них.

Дана сутність буде містити наступні поля:

1. RegionId
2. Name
3. Size
4. DangerStatus

RegionId – унікальний ідентифікатор області. Використовується як primary key.

Name - назва області.

Size - вказує на розмір області.

DangerStatus – вказує на статус небезпеки в області.

VolunteerCenter

VolunteerCenter – це сутність, що описує волонтерський центр, який зареєстрований за певною адресою.

Таблиця містить наступні поля:

1. VolunteerCenterId
2. Name
3. RegionId
4. Address

VolunteerCenterId – унікальний ідентифікатор волонтерського центру. Використовується як primary key.

Name – поле, що містить назву волонтерського центру.

RegionId – унікальний ідентифікатор області. Використовується як foreign key з посиланням на **Regions** таблицю.

Address – поле, містить інформацію про місце реєстрації волонтерського центру.

VolunteerCenterCategories

VolunteerCenterCategories – це сутність, що зв'язує види можливих гуманітарних допомог внутрішньо переміщеним особам з конкретними волонтерськими центрами. Таким чином, завдяки зв'язку багато до багатьох, один волонтерський центр може мати безліч категорій можливої допомоги.

Містить наступні поля:

1. Id
2. VolunteerCenterId
3. HelpCategoryId
4. ResponsiblePersonId

Id –унікальний ідентифікатор запису в таблиці. Є ключем primary key.

VolunteerCenterId – унікальний ідентифікатор, є ключемо foreign key з посиланням на таблицю **VolunteerCenter**

HelpCategoryId – унікальний ідентифікатор**,** є ключем foreign key з посиланням на таблицю **HelpCategory**

ResponsiblePersonId – унікальний ідентифікатор особи, яка є відповідальна за даний вид допомоги в даному волонтерському центрі. Є ключем foreign key з посиланням на таблицю **People**

HelpCategory

HelpCategory – таблиця, в якій зберігаються види гуманітарних допомог(їжа, медикаменти, засоби особистої гігієни, тощо).

Дана сутність буде містити наступні поля:

1. HelpCategoryId
2. CategoryName
3. Description

HelpCategoryId – унікальний ідентифікатор конкретного виду допомоги. Є ключем primary key.

CategoryName - поле для назви даної допомоги.

Description – поле, що описує вид допомоги.

Consumes

Consumes – таблиця, де зберігається інформація про отримання допомоги особою.

Містить такі дані в таблиці:

1. ConsumeId
2. PersonId
3. VolunteerCenterId
4. HelpCategoryId
5. Date
6. Weight

ConsumeId – унікальний ідентифікатор конкретного запису отримання допомоги. Є ключем primary key.

PersonId - 16-ти бітне унікальне поле, що буде генеруватися базою даних для цілісності та унікальності та містить унікальний ідентифікатор користувача. Є ключем foreign key з посиланням на таблицю **People.**

VolunteerCenterId – унікальний ідентифікатор, є ключемо foreign key з посиланням на таблицю **VolunteerCenter.**

HelpCategoryId – унікальний ідентифікатор**,** є ключем foreign key з посиланням на таблицю **HelpCategory.**

Date –Дата отримання допомоги.

Weight – Використовується для збереження кількості отриманої допомоги.

2.6 Моделювання предметної області на фізичному рівні.

Щоб зрозуміти, як ця інформація може бути реалізована в базі даних, давайте розглянемо програмні засоби, доступні для реалізації цього моделювання. Цей процес обговорюється в наступному підрозділі.

2.6.1 Переваги системи керування базами даних MS SQL.

SQL Server — це система керування базами даних, розроблена Microsoft. Це сервер даних, який в основному обслуговує дані у відповідь на запити від інших програм, ці запити можуть надходити з інших серверів або через мережу[5]. Використовувана мова запитів – Transact-SQL, створена спільно між Microsoft і Sybase. Transact-SQL — це стандартизована версія стандарту ANSI/ISO для мови структурованих запитів SQL із додатковими функціями. Він використовується як для малих, так і для великих баз даних, а також для баз даних усіх розмірів між ними. Протягом десятиліть вона успішно конкурувала з іншими системами керування базами даних, що і продовжує робити.

Є практично необмежена кількість компаній, включаючи Microsoft, що надають програмне забезпечення, яке полегшує розробку бізнес-додатків за

допомогою баз даних Microsoft SQL Server. Крім того, SQL Server 2005 містить Microsoft .NET Common Language Runtime (CLR), що дозволяє програмам, написаним на мовах платформи .NET (таких як VB.NET або C#), реалізовувати збережені процедури та різні функції.

Враховуючи таку велику популярність та підтримку даного продукту, був використаний MS SQL для нашої курсової роботи. Збереження бази даних відбувається в хмарних технологіях Azure SQL Server. Для адміністрування використовується веб-портал, на якому є широкий спектр налаштування, від безпеки, до аналітики використання.[14]

Рис. 2.6.1.1 (Веб-портал для роботи з сервером для бази даних)

2.6.2 Переваги інструмента для адміністрування баз даних Azure Data Studio

Azure Data Studio призначений для того, щоб зосередитися на функціональних можливостях, які найчастіше використовують розробники MS SQL. Рішення пропонує додаткові можливості, доступні як необов'язкові розширення. Воно створено для фахівців з обробки даних, які використовують

SQL Server і бази даних Azure у локальному середовищі або багатохмарних середовищах.[10]

Використовуйте процеси, які можна легко розширити, за допомогою сучасного інструменту аналізу даних, який використовує лише необхідні функції. Використовуйте повнофункціональний редактор запитів, створюйте власні блокноти, вбудовану підтримку Git і зручний термінал. Додавайте та видаляйте функції, щоб створити найефективніший інструмент для виконання завдання. Використовуйте розширення SQL Database Projects для створення SQL Server. Налаштуйте своє середовище відповідно до робочих процесів, які ви використовуєте найчастіше.[10]

Також великою перевагою даного інструменту є широкий спектр підтримуваних операційних систем, таких як: MS Windows, MacOS та Linux. Він також має розширену підтримку SQL, PowerShell, Python, KQL, Apache Spark і PySpark для надсилання запитів і керування SQL Server, PostgreSQL і Azure Data Explorer.

Простий, інтуїтивний та швидкий дизайн полегшують зконцентруватися на вирішенні завдання

29

Рис. 2.6.2.1 (Інтерфейс Azure Data Studio)

2.6.3. Опис структури бази даних та зв'язків між таблицями

Після вивчення логічного рівня бази даних під час попередньої роботи в цьому розділі інструмент Azure Data Studio буде використано для розробки таблиць і зв'язків бази даних.

В результаті пройдених кроків вище, була створена фізична модель кожного запису в нашій базі даних за допомогою SQL. Ці дані були використані для відображення, що складаються з команд SQL.

1. Таблиця People має інформацію про особу та має один ключ.
    - PersonId - який є унікальним. Є 128-бітний тип даних, який генерується на стороні бази даних, і є унікальним.
    - FirstName – NVARCHAR(50), символьний масив з лімітом в 50 символів. Не може бути null.

- SecondName - NVARCHAR(50), символьний масив з лімітом в 50 символів. Не може бути null.
- Email - NVARCHAR(320), символьний масив з лімітом в 320 символів. Не може бути null.
- DateOfBirth – Тип Date(дата), не може бути null
- Gender – тип bit, стать особи

Створюється наступною SQL командою:

```
CREATE TABLE [dbo].[People]
(
[PersonId] UNIQUEIDENTIFIER PRIMARY KEY default NEWID(),
[FirstName] NVARCHAR(50) NOT NULL,
[SecondName] NVARCHAR(50) NOT NULL,
[Email] NVARCHAR(320) NOT NULL,
[DateOfBirth] DATE NOT NULL,
[Gender] BIT NOT NULL
);
GO
```

Та в Azure Data Studio має наступний вигляд:

Рис. 2.6.3.1 (Таблиця Consumes)

2. Таблиця PeopleRegistrations має три ключі, та наступні поля:
   - RegistrationId - яке є унікальним. 128-бітний тип даних, який генерується на стороні бази даних, і є унікальним.
   - PersonId - 128-бітний тип даних, foreign key, що посилається на таблицю ключ **PersonId** з таблиці **People**
   - ServiceCenterId - 128-бітний тип даних, foreign key, що посилається на таблицю ключ **Id** з таблиці **ServiceCenter**
   - RegistrationDate – тип даних date(дата)

31

створюється наступною SQL командою:

```
CREATE TABLE [dbo].[PeopleRegistrations]
(
[RegistrationId] UNIQUEIDENTIFIER PRIMARY KEY default NEWID(),
[PersonId] UNIQUEIDENTIFIER NOT NULL FOREIGN KEY REFERENCES People(PersonId),
[ServiceCenterId]    UNIQUEIDENTIFIER NOT NULL FOREIGN    KEY REFERENCES
ServiceCenter(Id),
[RegistrationDate] DATE NOT NULL
);
GO
```

Та в Azure Data Studio має наступний вигляд:

Рис. 2.6.3.2 (Таблиця PeopleRegistrations)

3. Таблиця ServiceCenter має два ключі, та наступні поля:
    - Id - яке є унікальним. 128-бітний тип даних, який генерується на стороні бази даних, і є унікальним.
    - RegionId - 128-бітний тип даних, foreign key, що посилається на таблицю ключ **RegionId** з таблиці **Regions**
    - SettlementName – NVARCHAR(50), символьний масив з лімітом в 50 символів. Не може бути null.

створюється наступною SQL командою:

```
CREATE TABLE [dbo].[ServiceCenter]
(
[Id] UNIQUEIDENTIFIER PRIMARY KEY default NEWID(),
[RegionId] UNIQUEIDENTIFIER NOT NULL FOREIGN KEY REFERENCES Regions(Id),
[SettlementName] NVARCHAR(50) NOT NULL
);
GO
```

Та в Azure Data Studio має наступний вигляд:

32

Рис. 2.6.3.3 (Таблиця ServiceCenter)

4. Таблиця Regions має два ключі, та наступні поля:
    - Id - яке є унікальним. 128-бітний тип даних, який генерується на стороні бази даних, і є унікальним.
    - DangerStatusId - 128-бітний тип даних, foreign key, що посилається на таблицю ключ **Id** з таблиці **DangerStatus**
    - Name – NVARCHAR(50), символьний масив з лімітом в 50 символів. Не може бути null.

створюється наступною SQL командою:

```
CREATE TABLE [dbo].[Regions]
(
[Id] UNIQUEIDENTIFIER PRIMARY KEY default NEWID(),
[Name] NVARCHAR(50) NOT NULL,
[DangerStatusId]    UNIQUEIDENTIFIER NOT NULL FOREIGN    KEY REFERENCES
DangerStatus(Id)
);
GO
```

Та в Azure Data Studio має наступний вигляд:

Рис. 2.6.3.4 (Таблиця Regions)

5. Таблиця VolunteerCenter має два ключі, та наступні поля:

- VolunteerCenterId - яке є унікальним. 128-бітний тип даних, який генерується на стороні бази даних, і є унікальним.
- RegionId - 128-бітний тип даних, foreign key, що посилається на таблицю ключ **Id** з таблиці **Regions**
- Name – NVARCHAR(50), символьний масив з лімітом в 50 символів. Не може бути null.

створюється наступною SQL командою:

```
CREATE TABLE [dbo].[VolunteerCenter]
(
[VolunteerCenterId] UNIQUEIDENTIFIER PRIMARY KEY default NEWID(), -- Primary Key column
[Name] NVARCHAR(50) NOT NULL,
[RegionId] UNIQUEIDENTIFIER FOREIGN KEY REFERENCES Regions(Id),
[Address] TEXT NOT NULL
);
GO
```

Та в Azure Data Studio має наступний вигляд:

Рис. 2.6.3.5 (Таблиця VolunteerCenter)

6. Таблиця VolunteerCenterCategories має чотири ключі, та наступні поля:
   - Id - яке є унікальним. 128-бітний тип даних, який генерується на стороні бази даних, і є унікальним.
   - VolunteerCenterId - 128-бітний тип даних, foreign key, що посилається на таблицю ключ **VolunteerCenterId** з таблиці **VolunteerCenter**
   - HelpCategoryId - 128-бітний тип даних, foreign key, що посилається на таблицю ключ **HelpCategoryId** з таблиці **HelpCategory**

- ResponsiblePersonId - 128-бітний тип даних, foreign key, що посилається на таблицю ключ **PersonId** з таблиці **People**

створюється наступною SQL командою:

```
CREATE TABLE [dbo].[VolunteerCenterCategories]
(
[Id] UNIQUEIDENTIFIER PRIMARY KEY default NEWID(),
[VolunteerCenterId] UNIQUEIDENTIFIER FOREIGN KEY REFERENCES
VolunteerCenter(VolunteerCenterId),
[HelpCategoryId] UNIQUEIDENTIFIER FOREIGN KEY REFERENCES
HelpCategory(HelpCategoryId),
[ResponsiblePersonId] UNIQUEIDENTIFIER FOREIGN KEY REFERENCES People(PersonId),
);
GO
```

Та в Azure Data Studio має наступний вигляд:

Рис. 2.6.3.6 (Таблиця VolunteerCenterCategories)

7. Таблиця HelpCategory має один ключ, та наступні поля:
    - HelpCategoryId - яке є унікальним. 128-бітний тип даних, який генерується на стороні бази даних, і є унікальним.
    - CategoryName - NVARCHAR(50), символьний масив з лімітом в 50 символів. Не може бути null.
    - Description – текстовий типу даних.

створюється наступною SQL командою:

```
CREATE TABLE [dbo].[HelpCategory]
(
[HelpCategoryId] UNIQUEIDENTIFIER PRIMARY KEY default NEWID(),
[CategoryName] NVARCHAR(50) NOT NULL,
[Description] TEXT NOT NULL
);
GO
```

35

Та в Azure Data Studio має наступний вигляд:

Рис. 2.6.3.7 (Таблиця HelpCategory)

8. Таблиця Consumes має чотири ключі, та наступні поля:
    - Id - яке є унікальним. 128-бітний тип даних, який генерується на стороні бази даних, і є унікальним.
    - PersonId - 128-бітний тип даних, foreign key, що посилається на таблицю ключ **PersonId** з таблиці **People**
    - HelpCategoryId - 128-бітний тип даних, foreign key, що посилається на таблицю ключ **HelpCategoryId** з таблиці **HelpCategory**
    - VolunteerCenterId - 128-бітний тип даних, foreign key, що посилається на таблицю ключ **VolunteerCenterId** з таблиці **VolunteerCenter**

створюється наступною SQL командою:

```
CREATE TABLE [dbo].[Consumes]
(
[Id] UNIQUEIDENTIFIER PRIMARY KEY default NEWID(),
[PersonId] UNIQUEIDENTIFIER NOT NULL FOREIGN KEY REFERENCES People(PersonId),
[VolunteerCenterId]   UNIQUEIDENTIFIER NOT NULL FOREIGN   KEY REFERENCES
VolunteerCenter(VolunteerCenterId),
[HelpCategoryId]   UNIQUEIDENTIFIER NOT NULL FOREIGN   KEY REFERENCES
HelpCategory(HelpCategoryId)
);
GO
```

Та в Azure Data Studio має наступний вигляд:

36

Рис. 2.6.3.8 (Таблиця Consumes)

3. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

В попередньому розділі був розглянутий процес створення бази даних. Тепер потрібно використати цю базу даних для розробки як візуальної, так і логічної частин нашої інформаційної системи для організації роботи.

Клієнтська частина обробляє дані та представляє їх користувачеві у візуальному вигляді. У цьому розділі інформаційної системи буде описано, як будується інтерфейс.

Серверна частина обробляє великі порції інформації, дає можливість зберігати, транфсормувати, видаляти та оновлювати дані. Крім того, серверна частина підключаються до баз даних і виконує обробку даних.[1]

Завершимо проект опублікувавши його в хмарному середовищі, щоб зробити доступним для перегляду.

3.1 Опис інтерфейсу та функціональних можливостей програмної реалізації

Для створення інтерфейсів нашої інформаційної системи нам потрібно використовувати такі технології, як HTML, Css і Javascript. Нам також потрібен серверна частина проекту, оскільки дані з бази даних повинні бути опрацьовані. Для цього є чудова технологія ASP.NET Core MVC.

Після ретельного опрацювання бази даних, використовуючи вищезгадані інструменти, можна приступити до створення застосунку, щоб взаємодіяти з даними.

Функціоналу багато, тому сконцентруємося на основному. Перша сторінка, на яку хочеться звернути увагу – перелік всіх зареєстрованих тимчасово переміщених осіб. В ідеальному випадку, потрібна інтеграція з реєстром сервісних центрів, проте наразі дані про реєстрацію особи в тому чи іншому сервісному центрі зберігається в нашій базі даних.

1 https://www.educative.io/answers/client-side-vs-server-side

Рис. 3.1.1 (Список зареєстрованих осіб)

Наступним важливим функціоналом буде перегляд історії реєстрацій особи в різних сервісних центрах з важливою інформацією, для прикладу: дата та область реєстрації. Щоб зробити це, використовуємо випадаючий список “Опції”, після чого зможемо отримати перелік можливих операцій над даною особою.

Рис. 3.1.2 (Можливі дії по особі)

Для перегляду історії реєстрацій в сервісних центрах, потрібно натиснути на опцію “Історія реєстрацій”, після чого нам відкриється вікно з потрібно інформацією.

Рис. 3.1.3 (Історія реєстрацій особи)

Ця інформація збігається з актуальною інформацією іншої сторінки, що показує всі можливі сервісні центри, посортовані по областях. Дуже важливий момент для людини, яка видає гуманітарну допомогу – побачити реальний стан ситуації, звідки особа виїхала. Тому тут є стан, в якому перебуває область. На даному етапі це може бути: спокійно, лінія фронту, звільнено.

40

Рис. 3.1.4 (Відкриті сервісні центри)

Якщо потрібно змінити місце реєстрації особи, користувач повинен перейти до попередньої сторінки, та натиснути на випадаючий список “Опції”, після чого користувач зможемо виконати опцію “Зареєструвати”. Виглядає це наступним чином

Рис. 3.1.5 (Реєстрація особи в сервісному центрі)

Для вибору сервісного центру, є зручний елемент великої кількості можливих варіантів, тому користувач може вибрати один з 50 доступних сервісних центрів. Для збереження даних, достатньо лише натиснути “Зберегти”, після чого користувач буде направлений на сторінку з списком осіб. Для перевірки факту збереження інформації можна знову відкрити одне з вікон з інформацією про особу.

Рис. 3.1.6 (Список сервісних центрів)

А далі, саме цікаве. Основним функціоналом даної інформаційної системи є можливість видачі гуманітарної допомоги. Для цього переходимо в розділ “Волонтерська робота”, та натискаємо “Волонтерські центри”. В даній таблиці користувач зможе побачити список офіційних волонтерських центрів, місце знаходження та можливий вид допомоги.

Рис. 3.1.7 (Список офіційних волонтерських центрів)

Для надання допомоги вибераємо опцію “Перейти”, що відправить користувача на сторінку детальної інформації про волонтерський центр. Дана інформація також опрацьовується з бази даних, а саме: види гуманітарної допомоги, яку можуть надати, а також, список зареєстрованих осіб в сервісному центрі, що збігається з реєстрацією даного волонтерського центру.

Рис. 3.1.8 (Список доступних осіб для конкретного волонтерського центру)

В на даній сторінці в нас є можливість вибрати особу та надати допомогу. Для цього, в опціях користувач вибирає “Надати допомогу”, після чого відкриється нове вікно. В даному вікні користувач вибирає категорію, яка доступна його волонтерському центру, та натискаємо “Надати допомогу”.

Рис. 3.1.9 (Інтерфейс надання допомогу)

Така дія збереже факт надання допомоги в базу даних, з датою, особою, волонтерським центром та категорією допомоги. Знову ж таки, переглянути можна використавши опцію “Отримана допомога”, після чого користувач побачить історію отримання всіх допомог даною особою.

Рис. 3.1.10 (Огляд отриманої допомоги особою)

3.2 Забезпечення обробки та корегування даних в інформаційній системі.

Опрацювання даних, які користувач бачить на сторінках сервісу обліку видачі гуманітарної допомоги відбувається на серверній частині з використанням технології asp.net core mvc. З'єднання з нашою базою даних опрацьовує Entity Framework Core. Це фреймворк, що забезпечує організацію нашого коду серверної частини чистою, мінімалістичною, також передбачає ряд потенційних безпекових загроз, для прикладу SQL Injection.

Створимо об'єкт з ім'ям DbSubjectContext, що буде основним об'єктом, який надає нам доступ до наших сутностей з таблиць.

45

Рис. 3.2.1 (Сутності таблиць Entity Framework)

В рис(3.2.1) вище можна побачити колекції DbSet різних сутностей, які відповідають таблицям баз даних. Для прикладу, Person виглядає наступним чином

Рис. 3.2.2 (Сутність Person в Entity Framework)

В даному випадку, сутність Person містить такі поля, як:

1. Id, цілочисельний тип.
2. FirstName, текстовий тип.
3. SecondName, текстовий тип.
4. Email, текстовий тип.
5. DateOfBirth, тип дата.
6. Gender, текстовий тип.
7. І також віртуальні колекції, які полегшують вибірку даних.

Для прикладу, колеція Consumes зберігає інформацію про факт отримання особою допомоги. Тому, дана віртуальна колекція та інші дозволять нам зробити наступний запит в базу даних.

```
SELECT [c].[Id], [c].[Date], [h].[Id], [h].[CategoryName] AS [Name], [h].[Description], [v].[Id], [v].[Name], [v].[Address]
FROM [Consumes] AS [c]
INNER JOIN [HelpCategory] AS [h] ON [c].[HelpCategoryId] = [h].[Id]
INNER JOIN [VolunteerCenter] AS [v] ON [c].[VolunteerCenterId] = [v].[Id]
WHERE [c].[PersonId] = @__id_0
```

І результатом такого запиту, є інформація, яка потім використовується на даній сторінці.

47

Рис. 3.2.3 (Історія отриманої допомги особою)

Також для оновлення інформації, для прикладу, в таблиці VolunteerCenter опрацьовуємо наступний параметризований запит:

```
UPDATE [VolunteerCenter] SET [Address] = @p0
    WHERE [Id] = @p1;
```

А для реєстрації особи в певному сервісному центрі наступний параметризований запит

```
INSERT INTO [PeopleRegistrations] ([PersonId], [RegistrationDate], [ServiceCenterId])
    VALUES (@p0, @p1, @p2);
```

3.3 Публікація результатів обробки даних інформаційної системи в локальній та глобальній мережі.

Для публікації даної аплікації було використано наступні інструменти та технології: Azure Web App Service, Azure Sql Service, Github, Github Actions.

База даних зберігається в хмарному середовищі Azure portal, з використанням сервісу Azure Sql Service. Для налаштування є веб-застосунок, що надає доступ до SQL Server та можливість створювати/оновлювати бази даних. Також можемо бачити інформацію про навантаження на наш сервер.

Рис. 3.3.1 (Інформація навантаженості серверу бази даних)

Прогармний код нашої інформаційної системи зберігається в репозиторії github. Github actions надає можливість виконувати певні дії, коли публікуються нові зміни. Для прикладу, в даній роботі Github є налаштованим автоматично перевіряти наявність оновлень, збирати наш проект та розгортати на зовнішній сервер, який використається для зовнішього доступу в разі наявних оновлень.

Рис. 3.3.2 (Інтерфейс Github actions)

Автоматичне розгортання відбувається на Azure Web App Service, який в своїй сутності є контейнером, що здатний опрацьовувати http запити, та направляти на наш застосунок. Завдяки такій успішній автоматизації, є завжди оновлена версія проекту застосунку за адресою https://temporarymovedpeople.azurewebsites.net/

3.4 Рекомендації щодо впровадження та обгрунтування ефективності.

Є досить великий спектр можливого функціоналу, адже потенціал такої інформаційної системи дуже великий.

Основні, які можна було б реалізувати, щоб вже можна було б використовувати на постійні основі:

1. Автентифікація та авторизація: Firebase надає готові інструменти для реалізації системи аутентифікації користувачів. За допомогою Firebase

Authentication можна швидко налаштувати різні методи автентифікації, такі як електронна пошта та пароль, соціальні мережі, номер телефону тощо. Це спрощує процес управління користувачами та забезпечує безпеку доступу до функціональності сервісу.

2. Для кращого модерування такою складною системою пропоную створити багаторівневу модерацію, тому що ідея полягає в централізації даних на рівні держави. На мою думку, потрібні наступні ролі у всіх відділах:
    a. Адміністратор
    b. Модератор
    c. Користувач
    d. Оглядач

    Також відділами мають бути:
    e. Відділ модерації волонтерських центрів(можливо в руках місцевої влади)
    f. Волонтерський центр(кожен волонтерський центр є окремим відділом)
3. Покращити дизайн та зручність вводу даних.
4. Створити інтеграцію з різними відкритими державними реєстрами.

ВИСНОВКИ

Люди завжди були готові прислухатися до людей під час війни. Багато людей таким чином допомагали українцям — і вони не залишали без уваги ситуацію. Багато країн, організацій і людей надають допомогу. Але потрібно відстежувати допомогу, яку той чи інший волонтерський центр отримує, адже рано чи пізно – доведеться дати звіт.

Хоч і у перші дні війни, волонтерські центри швидко реалізувовули всі ресурси, які в них вливалися, тим не менш, зараз час, коли потрібно їх відстежувати. Планувати та звітувати про результати своїх зусиль. Крім того, враховуючи вимоги уряду від волонтерських центрів реєструвати ресурси, які вони приймають, примушує ретельно віднестися до даної проблеми.

З огляду на вказану вище проблему нами створено інформаційну систему, вигляд якої, міг би збільшитися у функціоналі, щоб використовувати ресурси максимально ефективно.

В роботі проаналізовано сучасні інструменти для розробки інформаційних систем, встановлені переваги та обґрунтовано доцільність застосування Azure SQL Server як серверу бази даних, ASP.NET Core 6 MVC для інформаційної системи, яка може обробляти дані та дає доступ користувачу до веб-інтерфейсу, та також інфраструктурні інструменти, такі як Github Actions та Azure Web App Service.

Ці технології є поширеними та популярними, що потрібно для такого проекту щоб полегшити пошук інженерів для розробки та мінімізувати час, витрачений на розробку.

А для нас, найважливіше зробити все можливе для забезпечення найшвидшої перемоги. А підтримка волонтерського руху, це дуже вагомий вклад в нашу перемогу.

ВИКОРИСТАНА ЛІТЕРАТУРА

1. Закон України Про благодійну діяльність та благодійні організації, https://zakon.rada.gov.ua/laws/show/5073-17#Text
2. Закон України про гуманітарну допомогу, стаття 1, https://zakon.rada.gov.ua/laws/show/1192-14#Text
3. Інструментарій для створення діаграм, https://app.diagrams.net
4. Інформаційний портал Visure, https://visuresolutions.com/uk/blog/requirements-specification/
5. Інформаційний портал wiki, https://en.wikipedia.org/wiki/Microsoft_SQL_Server
6. Переваги та недоліки програми «1С: Бухгалтерія», https://stimul.kiev.ua/articles.htm?a=perevagi_ta_nedoliki_programi_laquo1s_bukhgalteriyaraquo
7. Специфікація вимог до програмного забезпечення (SRS): Поради та шаблон, https://visuresolutions.com/uk/software-requirement-specification-srs-tips-template/
8. Функціональні можливості ПЗ 1С:Підприємство, http://1c.ua/ua/v8/RegionalSolutions_UA_ZUP.php
9. Що таке специфікація вимог: визначення, найкращі інструменти та методи, https://visuresolutions.com/uk/blog/requirements-specification/
10. Azure Data Studio, https://azure.microsoft.com/ru-ru/products/data-studio/#features,
11. Azure Web App tutorial, https://www.javatpoint.com/azure-web-app
12. Database – User and Group tables [Електронний ресурс] – Режим доступу до ресурсу: https://developer.atlassian.com/server/jira/platform/database-user-and-group-tables/.
13. Download and install Azure Data Studio, https://learn.microsoft.com/en-us/sql/azure-data-studio/download-azure-data-studio?view=sql-server-ver16
14. Explore Azure SQL database services, https://azure.microsoft.com/en-us/products/azure-sql/
15. GitHub Actions, complete documentation, https://docs.github.com/en/actions
16. Microsoft Azure on Wikipedia, https://en.wikipedia.org/wiki/Microsoft_Azure

17. The Basics of Database Normalization, https://www.lifewire.com/database-normalization-basics-1019735
18. What is Normalization in DBMS (SQL)?, https://www.guru99.com/database-normalization.html
19. What is a relational database?, https://www.techtarget.com/searchdatamanagement/definition/relational-database

ДОДАТКИ

Додаток А

1. Посилання на зовнішню адресу інформаційної системи, http://temporarymovedpeople.azurewebsites.net

Додаток В

Частина серверної частина, як опрацьовує запити про особу.

```
public class PeopleController : Controller
{
    private readonly IPeopleService _peopleService;
    private readonly IRegionsService _regionsService;
    private readonly IServiceCenterService _serviceCenterService;

    public PeopleController(
        IRegionsService regionsService,
        IPeopleService peopleService,
        IServiceCenterService serviceCenterService)
    {
        _regionsService = regionsService;
        _peopleService = peopleService;
        _serviceCenterService = serviceCenterService;
    }

    public IActionResult Index()
    {
        PeopleViewModel model = new PeopleViewModel(){
            People = _peopleService.GetAll().ToList()
        };
        return View(model);
    }

    public IActionResult Person(int id) => View(_peopleService.GetPersonViewModel(id));

    public IActionResult EditPerson(int id) =>
      View(_peopleService.GetPersonViewModel(id));

    [HttpGet]
    public IActionResult ConfirmDeletePerson(int id) =>
      View(_peopleService.GetPersonViewModel(id));

    [HttpDelete]
    public IActionResult DeletePerson(int id)
    {
        try{
            _peopleService.DeletePerson(id);
        }
```

55

```
        catch (DbUpdateException exception)
        {
            return RedirectToAction("PersonIsUsed", "People");
        }

        return RedirectToAction("Index", "People");
    }

    public IActionResult PersonIsUsed() => View();

    public IActionResult AddPerson() => View();

    [HttpPost]
    public IActionResult AddPerson(PersonViewModel person)
    {
        _peopleService.AddPerson(person);
        return RedirectToAction("Index", "People");
    }

    [HttpPost]
    public IActionResult EditPerson(PersonViewModel person)
    {
        _peopleService.UpdatePerson(person);
        return RedirectToAction("Index", "People");
    }

    public IActionResult RegisterPerson(int personId){
        var lastRegistration = _peopleService.GetPersonRegisters(personId);
        var allServiceCenters = _serviceCenterService.GetAll().Select(x => new
      ServiceCenterViewModel(){
                Id = x.Id,
                SattlementName = x.SattlementName
            }).ToList();
        RegisterViewModel model = new RegisterViewModel(){
            PersonId = personId,
            Name = String.Format("{0} {1}", lastRegistration.Person.FirstName,
      lastRegistration.Person.SecondName),
            ServiceCenterId =
      lastRegistration.Registers.LastOrDefault()?.ServiceCenterId,
            ServiceCenters = allServiceCenters
        };
        return View(model);
    }

    [HttpPost]
    public IActionResult RegisterPerson(int personId, RegisterRequestModel model)
    {
        model.PersonId = personId;
        _peopleService.RegisterPerson(model);
        return RedirectToAction("Index", "People");
    }
```

```
    public IActionResult PersonConsumes(int id) =>
      View(_peopleService.GetPersonConsumes(id));
    public IActionResult GetPersonRegistrations(PersonViewModel person) =>
      View(_peopleService.GetPersonRegisters(person.Id));

    public IActionResult PersonRegistrations() => View();

    [ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)]
    public IActionResult Error()
    {
        return View(new ErrorViewModel { RequestId = Activity.Current?.Id ??
      HttpContext.TraceIdentifier });
    }
}
```

Додаток С

Частина серверної частини, як генерує HTML, CSS та JS перед відправленням на сторону клієнта(браузера).

```
@model PeopleViewModel;
@{
    ViewData["Title"] = "Всі зареєстровані особи";
}

<div class="text-center">
    <h1 class="display-8">Всі зареєстровані особи</h1>
    <div class="people">
        <table class="table">
            <thead>
                <tr>
                    <th scope="col">Ім'я</th>
                    <th scope="col">Прізвище</th>
                    <th scope="col">Поштова скринька</th>
                    <th scope="col">День народження</th>
                    <th scope="col">Стать</th>
                    <th scope="col">Дії</th>
                </tr>
            </thead>
            <tbody>
                @foreach (var person in Model.People)
                {
                    <tr>
                        <td>@person.FirstName</td>
                        <td>@person.SecondName</td>
                        <td>@person.Email</td>
                        <td>@person.DateOfBirth.ToShortDateString()</td>
                        <td>@person.Gender</td>
                        <td>
                            <div class="dropdown">
                                <button class="btn btn-secondary dropdown-toggle"
type="button" id="dropdownMenuButton1" data-bs-toggle="dropdown" aria-expanded="false">
                                    Опції
```

```
                                </button>
                                <ul class="dropdown-menu" aria-
labelledby="dropdownMenuButton1">
                                    <li><a class="dropdown-item"
onclick="openPersonEdit(@person.Id)">Редагувати</a></li>
                                    <li><a class="dropdown-item"
onclick="openPersonDetails(@person.Id)">Деталі</a></li>
                                    <li><a class="dropdown-item"
onclick="openPersonConsumes(@person.Id)">Отримана допомога</a></li>
                                    <li><a class="dropdown-item"
onclick="openPersonRegistrations(@person.Id)">Історія реєстрацій</a></li>
                                    <li><a class="dropdown-item"
onclick="openPersonRegister(@person.Id)">Зареєструвати</a></li>
                                    <li><a class="dropdown-item"
onclick="confirmDelete(@person.Id)">Видалити</a></li>
                                </ul>
                            </div>
                        </td>
                    </tr>
                }
            </tbody>
        </table>
    </div>
</div>
<script type="text/javascript">
    function confirmDelete(id){
        $.ajax({
            type: "GET",
            url: "People/ConfirmDeletePerson",
            data: { "id": id },
            success: function (response) {
                $("#modal").find(".modal-content").html(response);
                $("#modal").modal('show');
            },
            failure: function (response) {
                alert(response.responseText);
            },
            error: function (response) {
                alert(response.responseText);
            }
        });
    }

    function openPersonConsumes(id){
        $.ajax({
            type: "GET",
            url: "People/PersonConsumes",
            data: { "id": id },
            success: function (response) {
                $("#modal").find(".modal-content").html(response);
                $("#modal").modal('show');
            },
            failure: function (response) {
```

```
            alert(response.responseText);
        },
        error: function (response) {
            alert(response.responseText);
        }
    });
}

function openPersonRegister(personId){
    $.ajax({
        type: "GET",
        url: "People/RegisterPerson",
        data: { "personId": personId },
        success: function (response) {
            $("#modal").find(".modal-content").html(response);
            $("#modal").modal('show');
        },
        failure: function (response) {
            alert(response.responseText);
        },
        error: function (response) {
            alert(response.responseText);
        }
    });
}

function openPersonDetails(id) {
    $.ajax({
        type: "GET",
        url: "People/Person",
        data: { "id": id },
        success: function (response) {
            $("#modal").find(".modal-content").html(response);
            $("#modal").modal('show');
        },
        failure: function (response) {
            alert(response.responseText);
        },
        error: function (response) {
            alert(response.responseText);
        }
    });
}

function openPersonRegistrations(id) {
    $.ajax({
        type: "GET",
        url: "People/GetPersonRegistrations",
        data: { "id": id },
        success: function (response) {
            $("#modal").find(".modal-content").html(response);
            $("#modal").modal('show');
        },
```

59

```
            failure: function (response) {
                alert(response.responseText);
            },
            error: function (response) {
                alert(response.responseText);
            }
        });
    }

    function openPersonEdit(id) {
        $.ajax({
            type: "GET",
            url: "People/EditPerson",
            data: { "id": id },
            success: function (response) {
                $("#modal").find(".modal-content").html(response);
                $("#modal").modal('show');
            },
            failure: function (response) {
                alert(response.responseText);
            },
            error: function (response) {
                alert(response.responseText);
            }
        });
    }
</script>
```

Схожість

Джерела з Бібліотеки 350

1	Студентська робота	ID файлу: 8562579	Навчальний заклад: V.I. Vernadsky Taurida National Univers	46 Джерело	1.55%
2	Студентська робота	ID файлу: 1004978431	Навчальний заклад: National University of Life and Envi	46 Джерело	1.87%
3	Студентська робота	ID файлу: 1014817933	Навчальний заклад: Taras Shevchenko National Universit	4 Джерело	1.43%
4	Студентська робота	ID файлу: 1014786054	Навчальний заклад: National University Ostroh Academy	3 Джерело	1.36%
5	Студентська робота	ID файлу: 1015302102	Навчальний заклад: National University of Water Manag	74 Джерело	1.4%
6	Студентська робота	ID файлу: 3277831	Навчальний заклад: Lviv Polytechnic National University	6 Джерело	0.49%
7	Студентська робота	ID файлу: 1005001533	Навчальний заклад: Taras Shevchenko National University of Kyiv		0.75%
8	Студентська робота	ID файлу: 1011481903	Навчальний заклад: National University of Water Manag	28 Джерело	0.72%
9	Студентська робота	ID файлу: 1015112178	Навчальний заклад: National Aviation University		0.65%
10	Студентська робота	ID файлу: 1014538439	Навчальний заклад: National University Ostroh Academy	5 Джерело	0.21%
11	Студентська робота	ID файлу: 1011573817	Навчальний заклад: Vasyl Stus Donetsk National University		0.12%
12	Студентська робота	ID файлу: 1114782	Навчальний заклад: Lviv Polytechnic National University	3 Джерело	0.13%
13	Студентська робота	ID файлу: 1002190629	Навчальний заклад: Taras Shevchenko National University of Kyiv		0.24%
14	Студентська робота	ID файлу: 1004153401	Навчальний заклад: National University of Water Manage	2 Джерело	0.28%
15	Студентська робота	ID файлу: 1008305519	Навчальний заклад: Lviv Polytechnic National University	14 Джерело	0.06%
16	Студентська робота	ID файлу: 1000095414	Навчальний заклад: National University of Water Manag	57 Джерело	0.24%
17	Студентська робота	ID файлу: 1009726236	Навчальний заклад: National Aviation University	3 Джерело	0.18%
18	Студентська робота	ID файлу: 106306	Навчальний заклад: National University of Life and Environmental Scie...		0.18%
19	Студентська робота	ID файлу: 5692536	Навчальний заклад: National University of Life and Environ	12 Джерело	0.15%
20	Студентська робота	ID файлу: 1010744240	Навчальний заклад: Ternopil Volodymyr Hnatiuk Nation	19 Джерело	0.13%

21	Студентська робота	ID файлу: 1015188660	Навчальний заклад: National Aviation University		0.13%
22	Студентська робота	ID файлу: 1015170921	Навчальний заклад: National Technical University of Ukraine "Kyi...		0.12%
23	Студентська робота	ID файлу: 1011350518	Навчальний заклад: Lviv Polytechnic National University	2 Джерело	0.12%
24	Студентська робота	ID файлу: 1013082490	Навчальний заклад: National University of Water Manage	5 Джерело	0.11%
25	Студентська робота	ID файлу: 1013071127	Навчальний заклад: Izmail State University of Humanities	2 Джерело	0.11%
26	Студентська робота	ID файлу: 1015119262	Навчальний заклад: National University of Life and Envir	2 Джерело	0.11%
27	Студентська робота	ID файлу: 1015240963	Навчальний заклад: National Technical University of Ukraine "Kyi...		0.11%
28	Студентська робота	ID файлу: 1000818595	Навчальний заклад: Ukrainian Catholic University	2 Джерело	0.09%
29	Студентська робота	ID файлу: 5990443	Навчальний заклад: National Technical University of Ukraine	5 Джерело	0.09%
30	Студентська робота	ID файлу: 1015280336	Навчальний заклад: Lviv Polytechnic National University		0.09%
31	Студентська робота	ID файлу: 1015277388	Навчальний заклад: Lviv Polytechnic National University		0.09%

Цитати

Цитати **7**

1 DeletePerson(id); } 55 catch (DbUpdateException exception) { return RedirectToAction("PersonIsUsed", "People"); } return RedirectToAction("Index", "People"); } public IActionResult PersonIsUsed() => View(); public IActionResult AddPerson() => View(); [HttpPost] public IActionResult AddPerson(PersonViewModel person) { _peopleService.

2 AddPerson(person); return RedirectToAction("Index", "People"); } [HttpPost] public IActionResult EditPerson(PersonViewModel person) { _peopleService.

3 GetPersonConsumes(id)); public IActionResult GetPersonRegistrations(PersonViewModel person) => View(_peopleService.

4 Id)); public IActionResult PersonRegistrations() => View(); [ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)] public IActionResult Error() { return View(new ErrorViewModel { RequestId = Activity.

5 **«Розробка інформаційної системи для обліку отримання допомоги тимчасово переміщених осіб»**

6 **«Благодійна діяльність — добровільна особиста та/або майнова допомога для досягнення визначених цим Законом цілей, що не передбачає одержання благодійником прибутку, а також сплати будь-якої винагороди або компенсації благодійнику від імені або за дорученням бенефіціара»**

7 **“Про благодійну діяльність та благодійні організації”»**