# 1. ВСТУП

**Слайд 3**

Темою бакалаврської кваліфікаційної роботи є розробка спеціалізованої системи розподілу навантаження між викладачами кафедри. Як відомо, кількість годин між працівниками освіти в університетах передбачені навчальним планом та чітко встановлені окремо для кожного викладача. Це пов’язано з тим, що відповідно до кількості годин викладання обраховується ставка працівника. Розподіл навантаження обчислюється представником кафедри ще до початку навчального процесу згідно з Інструкцією про порядок обчислення заробітної плати працівників освіти.

Саме на цьому етапі і виникає основна проблема процесу розподілу. Так як це виконує одна людина, то обчислення займають велику частку робочих годин самого ж працівника. До того ж, не виключено, що під час виконання такого трудомісткого завдання, спрацює людський фактор і призведе до помилки. Окрім того, протягом навчального процесу доводиться не одноразово повторювати цю ж роботу через регулярні заміни між викладачами, скорочення або прийняття на роботу нового працівника.

Описана у проєкті автоматизована система надає змогу швидкому та простому розрахунку розподілу годин навантаження між працівниками кафедри. По-перше, програмне рішення забезпечить надійність даних та зменшить вірогідність помилки під час обчислення. По-друге, система значно економить час самого працівника університету.

# 2. Аналі попередніх проєктних рішень та технічного завдання

## 2.1 Trello

**Слайд 4**

Trello – це неймовірний веб-інструмент для керування проектами та спільної роботи, який допоможе планувати проекти на одній платформі. Він неймовірно візуальний і дуже адаптивний, що полегшує ваше життя, коли над проектом працює кілька людей, особливо коли вони розкидані по різних місцях.

## 2.2 Jira

**Слайд 5**

Jira – це інструмент відстеження помилок, який, в основному, був розроблений австралійською компанією Atlassian у 2002 році.

Jira Core - це рішення для управніння проектами та завданнями.

## 2.3 ClickUp

**Слайд 6**

ClickUp – це хмарна робоча платформа для всіх типів і розмірів команд і компаній. Він поєднує важливі бізнес-додатки та централізує інформацію про копанію в єдине онлайн-рішення.

## 2.4 Todoist

**Слайд 7**

Todoist є одним із найуживаніших програм на сучаному ринку. Він наповнений чудовими функціями, які підвищують продуктивність, організовують завдання та встановлюють графіки, а також терміни.

## 2.5 Технічне завдання

**Слайд 8**

Спеціалізована програма розподілу навантаження між викладачами кафедри є унікальною. Ні один з вищевказаних подібних програмних рішень не підтримує потрібні функції. В програмному рішенні обов‘язоково повинні бути імплементовані наступні функції:

1) Форми для ручного заповнення

Форми для ручного заповнення даних потрібні для корегування або заповнення даних при не можливості імпорту з іншого джерела.

2) Кросплатформенність

Кросплатформенність потрібна для того щоб використовувати розроблене програмне рішення під керуванням будь якої операційної системи.

3) Зручний інтерфейс

Зручний інтерфейс потрібен для зручності використання продукту. Чим зручніший інтерфес, тим менше потрібно потрати часу на виконання завдання.

4) Валідація даних

Валідація даних потрібна для уникнення помилок повязаних з людським фактором.

1. Динамічна рекалькуляція розподілу навантаження між викладачами

Динамічна рекалькуляція розподулу навантаження між викладачами потрібна для зменшення часу виконання завдання а також для зменшення помилки в розрахунках. Попередньо всі дані перераховувалися в ручну.

1. Можливість експорту даних в csv, xlsx

Можливість експорту даних потрібна для подальшого їх використання деканатом, або іншими керуючими апаратами.

1. Проста СУБД

Проста СУБД потрібна для спрощення переносу даних.

Було проаналізовано 4 сервіси, які могли б вирішити поставлене технічне завдання. Всі перелічені сервіси є надлишково функціональні. В розглянутих програмних рішеннях нема функціоналу, поставленого в технічному завданні. Також більшість функціоналу є платним. Технічне завдання було проаналізовано. Ключові особливості створегого рішення: зручний інтерфейс, валідація даних, динамічна рекалькуляція та проста СУБД.

# 3. Вибір засобів для реалізації

**Слайд 9**

Всі використані технолої були вибрані опираючись на досвід використання в попередніх комерційних проектах. Технології показали максимальну швидкодію, мінімальний час на розробку а також легкість використання.

Короткий перелік технологій та місця їх використання:

1. Клієнтська частина

* TypeScript – об‘єктно-орієнтована мова програмування, яка компілюється в JavaScript. Виконується в браузері та допомагає анімувати веб-сторінку.
* Angular – фреймворк, який спрощує роботу з TypeScript і зменшує час, який потрібен на розробку програмного рішення.
* HTML – декларативна мова для опису веб-сторінок. Відповідає за наповнення web-сторінки.
* SCSS – декларативна мова, яка компілюється в CSS. Відповідає за відображення web-сторінки.

1. Серверна частина

* C# - флагманська мова програмування компанії Microsoft. Відповідає за обробку даних та взаємодію з базою даних.
* ASP.NET Core – фреймворк, який спрощує роботу з C#. В ньому імплементований функціонал для роботи з HTTP.
* EntityFramework Core – фреймворк який спрощує взаємодію з базою даних. API між базою даних і C# кодом.
* SQLite – компактна СУБД. Використвоується для збереження даних.

# 4. Випадки використання прогами та алгоритм роботи користувача

**Слайд 10**

В користувача є 11 випадків використання програмного продукту:

1. Створити проект

Проект це абстактна сутність, до якої можуть прив‘язуватися інші сутності. Він слугує одиницею розмежування роботи. Різні проекти можуть містити свої унікальні дисципліни, викладачів тощо.

1. Вибрати проект для роботи

Після вибору проекту користувач може створювати інші сутності. Вони будуть використовуватись для збереження і обрахунку даних.

1. Створити університет

Сутність універститету допомагає досягнути універсальності. В майбутньому можна використовувати аплікацію для різних кафедр та університетів.

1. Створити активність

Під активністю мається на увазі «Лаборатна», «Курсова», «Димпломна» робота тощо. Теж допомагає досягти універсальності.

1. Створити лектора

Сутність лектора потрібна для співставленням з реальним лектором.

1. Створити дисципліну

Сутність дисципліни потрібна для співставлянням з реальною дисципліною. Вона повинна містити кількість студентів, груп тощо.

1. Приєднати лектора до університету

Сутність зв‘язку Лектор-Університет уникає дублювання записів в базі даних. Один лектор може викладати в декількох університетах або на різних кафедрах.

1. Приєднати лектора до активності і дисципліни.

Сутність зв‘язку Лектор-Активність-Дисципліна теж уникає дублювання записів в базі даних. Один лектор може проводити як лабораторні так і лекції в декількох дисциплінах.

1. Запустити обрахунок даних

Запуск обрахунку даних розподіляє навантаження між викладачами опираючись на попередньо додані дані.

10) Експортувати дані

Експорт даних це міграція даних з бази даних до xlsx або csv для подальшого використання деканатом або іншими керуючими органами.

1. Переглянути логи

Перегляд логів це системна функція. В логах буде зберігатися інформація про роботу програми або помилки, які виникли під час роботи. Опираючись на логи буде простіше пояснити розробнику про проблеми, які могли виникнути під час експлуатації програмного продукту. Таким чином розробник витратить менше часу на виправлення помилок в коді програми.

# 5. Архітектура аплікації

**Слайд 12**

Покладаючись на висунуті технічні вимоги я побудував артіхектуру, яка складається з двох частин. Перша – клієнтська частина, друга – серверна частина. В майбутньому серверну частину можна розгорнути на будь-якому сервері або в хмарі. Внаслідок цього користувач може працювати з аплікацією з будь-якого пристрою, знаходячись в будь-якій точці світу де є доступ до Інтернету. Клієнтська частина(Браузер) обмінюється даними з REST API. Який в свою чергу оперує даними з бази даних. В базі даних зберігаються потрібні користувачу дані.

**Слайд 14**

Загалом було створено 3 інтерфейса, 2 абстрактні класи та 9 доменних класів. Всі доменні класи відповідають таблицям в базі даних, які були описані в розділі 3.1.1. Опис абстрактних класів та інтерфейсів:

1. IEntity

Інтерфейс для поліморфної поведінки доменних моделей. Містить ідентифікатор. Використовується в абстракному репозиторію для взаємодії з базою даних.

1. ICreateble

Інтерфейс для поліморфної поведінки доменних моделей. Містить дату створення запису в базі даних. Використовується для сортування по даті створення в абстрактному репозиторію.

1. IUpdateble

Інтерфейс для поліморфної поведінки доменних моделей. Містить дату оновлення записів в базі даних. Використовується для збереження дати останньої зміни доменної моделі.

1. BaseEntity

Абстрактний клас для уникання копіювання властивостей в доменних моделях.

1. BaseProjectRelatedEntity

Абстрактний клас для уникання копіювання навігаційних властивостей в доменних моделях повязаних з проектом.

**Слайд 15**

Було створено 1 сервіс та 1 шаблонний репозиторій, а також інтерфейси для них. Інтерфейси будуть використані для швидкої зміни реалізацій сервісів або репозиторіїв в разі потреби.

1. ILoggerService

Інтерфейс, в якому описані методи логування помилок.

1. DbLoggerService

Клас, який використовується для збереження логів в базі даних.

1. IRepository

Шаблонний інтерфейс, в якому описані методи для взаємодії з базою даних.

1. SQLiteRepository

Шаблонний клас, який використовується для взаємодії з базою даних.

**Слайд 16**

Контроллери служать для трансферу даних з бази даних до клієнтської частини. Було створено 11 класів, 3 з яких абстрактні.

1. Controller

Абстрактний шаблонний клас. Створений для базових CRUD операцій і зменшення кількості коду.

1. CollectionController

Абстрактний шаблонний клас. Створений для маніпуляції колекціями даних не повязаних з проектом.

1. ProjectRelatedCollectionController

Абстрактний шаблонний клас. Створений для маніпуляції колекціями даних повязаних з проектом.

Всі інші контроллери наслідуються від абстрактних і можуть перевантажувати методи цих контроллерів. Перевантаження залежить від особливостей реалізації кінцевого контролера.

**Слайд 17**

Для абстрагування та зменшення дубляжу коду було створено 6 допоміжних класів:

1. CUComponent

Клас використовується для зберігання базових даних компонетів. В ньому реалізовано методи обробки подій а також оновлення даних на веб сторінці. Цей клас забезпечує 2 базові операції: створення та оновлення даних.

1. OperationManager

Використовується для агрегації базовий операцій для колекцій даних. Такі як: оновлення, видалення і створення.

1. PaginatorI18n

Служиь для локалізації елементу управління «Paginator».

4) ProjectHanlder

Використовуэться для реактивного вибору проекту.

5) TableDataSource

Агрегує операцї, потрібні для відображення колекцій даних.

1. TableComponent

Базовий компонент для роботи з таблицями на клієнтській стороні.

**Слайд 18**

Сервіси були створені для зменшення дубляжу коду в компонентах. Наразі створено 3 сервіса:

1. CalculationService

Сервіс створений для обрахунку розподілу навантаження між викладачами кафедри.

1. EnumService

Створений для спрощення роботи з типом даних enum в TypeScript.

**Слайд 19**

Клієнти це окремий вид сервісів, які служать для комунікації клієнтської частини з серверною. Вони виконують HTTP запити до сервера і конвертують результат. Клієнти використовуються в сервісах та компонентах для спрощення комунікації з сервером.

# 6. Висновки

**Слайд 25**

В ході роботи над дипломним проектом була розроблена спеціалізована програма розподілу навантаження між викладачами кафедри. Проаналізоване існуюче програмне забезпечення. А саме: Trello, Jira, ClickUp, Todoist. Аналіз показав, що існуючі системи мають дуже багато переваг та недоліків. В існуючих програмних рішеннях не імплементовані всі функції вказані в завданні. Такі як: створення лекторів, університетів, активностей, дисциплін та зв‘язків між ними; експорт даних; обрахунок навантаження. Також розглянуті рішення є складними для вирішення частини завдань.

У процесі розробки було реалізовано всі необхідні функції вказані в технічному завданні. Побудовано зручну та універсальну структуру зв‘язків між сутностями. Використані сучасні технології, які полегшують розробку, зводять час розробки до мінімуму, максимізують швидкодію. Було застосовано кілька різних підходів до проектування програмного забезпечення. Спроектовано монолітну клієнт-серверну архітектуру на основі Angular та ASP.NET Core.

Розроблене програмне рішення дозволяє вводити необхідні дані, валідувати їх, обраховувати навантаження а також експортувати результати в будь-який зручний користувачеві формат. Аплікація зменшує час, який потрібно потратити на розподіл навантаження. Зводить до мінімуму можливість помилки. Вона дуже зручна у використанні і не потребує спеціальних навичок роботи з комп‘ютером.

Систему було налагоджено, протестовано та випробувано. Мануальні тести показали коректність роботи програми. Система є ефективною і вирішує всі задачі, які було прописані в технічному завданні. Програмний продукт показав хороші результати при тестуванні швидкодії. А саме: час запиту до API – 54 мс, час запиту до бази даних – 4 мс, час завантаження веб сторінки – 48 мс.

Потенційними користувачами програми є викладачі кафердри, яким кожного семестру потрібно було витрачати багато часу на розподіл навантаження.

Програмне рішення може бути використано на будь-які операційній системі ПК, має бути встановлено барвзер і пакети ASP.NET Core.