НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Навчально-науковий інститут атомної та теплової енергетики
Кафедра цифрових технологій в енергетиці

"На правах рукопису"
УДК_004.42_

«До захисту допущено»
Завідувач кафедри
_____ Наталія АУШЕВА
"___"_____2022р.

Магістерська дисертація

зі спеціальності - 122 Комп'ютерні науки
за освітньо-професійною програмою магістерської підготовки - Комп'ютерний моніторинг та геометричне моделювання процесів і систем

на тему «Моделювання та автоматизація планування навчального навантаження кафедри»

Виконав : студент _2_курсу, групи_ТР-11мп___
__Осипов Владислав Ігорович_____ _____
(прізвище, ім'я, по батькові) (підпис)

Науковий керівник професор, доцент, д.т.н._Шушура О.М. _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Консультант _____ _____ _____
(назва розділу) (вчені ступінь та звання, прізвище, ініціали) (підпис)

Рецензент_____ _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ - 2022

Національний технічний університет України
“ Київський політехнічний інститут ім. Ігоря Сікорського”

Навчально-науковий інститут атомної та теплової енергетики
Кафедри цифрових технологій в енергетиці
Рівень вищої освіти другий, магістерський
За освітньою програмою "Комп’ютерний моніторинг та геометричне моделювання процесів і систем"

Спеціальності 122 Комп’ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Наталія АУШЕВА
(підпис)

«____» _____2022р.

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ ДИСЕРТАЦІЮ СТУДЕНТУ

Осипову Владиславу Ігоровичу
(прізвище, ім’я, по батькові)

1. Тема дисертації «Моделювання та автоматизація планування навчального навантаження кафедри»

Науковий керівник Шушура Олексій Миколайович, д.т.н., доцент
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від “7”листопада 2022 року №_4067-с__

2. Строк подання студентом дисертації 7 грудня 2022року

3. Вихідні дані до роботи інформаційна технологія для автоматизації планування навчального навантаження кафедри

4. Перелік питань, які потрібно розробити дослідити процес планування навчального навантаження кафедри, спроектувати систему, розробити застосунок

5. Орієнтований перелік ілюстративного матеріалу приклади робочого навчального плану і плану навчального навантаження науково-педагогічних працівників, діаграми прецедентів, схема роботи архітектури системи, приклад інтерфейсу

6. Орієнтований перелік публікацій тези «Інформаційна технологія для автоматизації планування навчального навантаження кафедри»

7. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

8. Дата видачі завдання «_28_» жовтня 2021р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання магістерської дисертації	строки виконання етапів магістерської дисертації	Примітка
1	Отримання завдання	28.09.2021	
2	Практика	1.09.2022-26.10.2022	
3	Збір інформації	01.10.2021-05.03.2022	
4	Аналіз вимог завдання, розробка методів і засобів розв'язання поставленої задачі	06.03.2022-31.05.2022	
5	Розробка та тестування програмного продукту	02.06.2022-26.10.2022	
6	Виконання розділів дисертації	31.05.2022-15.11.2022	
7	Написання основних розділів автореферату	15.11.2022-18.11.2022	
8	Перевірка дисертації науковим керівником	19.11.2022	
9	Передзахист магістерської дисертації та допуск до захисту	23.11.2022	
10	Подання в електронному вигляді роботи та анотації на перевірку нормоконтролера	30.11.2022	
11	Подання магістерської дисертації рецензенту. Отримання рецензії	02.12.2022	
12	Подання пакету документів з магістерської дисертації та супутних до неї до захисту в ЕК	08.12.2022	
13	Захист магістерської дисертації	20.12.2022	

Студент _____ Осипов В.І.
(підпис) (прізвище та ініціали)

Науковий керівник _____ Шушура О.М.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Магістерська дисертація за темою «Моделювання та автоматизація планування навчального навантаження кафедри» виконана студентом кафедри цифрових технологій в енергетиці НН ІАТЕ Осиповим Владиславом Ігоровичем зі спеціальності 122 «Комп'ютерні науки» за освітньо-професійною програмою «Цифрові технології в енергетиці» та складається зі: вступу, 5 розділів («Постановка задачі автоматизації навчального навантаження кафедри», «Аналіз предметної області», «Опис програмної реалізації», «Робота користувача з системою», «Розробка стартап-проєкту»), висновків до кожного з цих розділів, загальних висновків, списку використаних джерел, який налічує 35 джерел. Загальний обсяг роботи 116 сторінок.

Актуальність теми. Планування навчального навантаження кафедри є дуже важливим процесом в роботі кафедри та сильно впливає на продуктивність як навчання студентів, так і на викладання дисциплін науково-педагогічними працівниками. Хоча системи для автоматизації даного виду діяльності існують, жодна з них не задовільняє повною мірою потреб користувачів. Це і стало причиною прийняття рішення про проведення дослідження на цю тему.

Метою дослідження є автоматизація планування навчального навантаження кафедри шляхом розробки спеціалізованої системи.

Завдання дослідження:

- аналіз процесу планування навчального навантаження кафедри;
- дослідження існуючих програмних рішень для автоматизації планування навчального навантаження кафедри;
- формування вимог до майбутньої системи;
- аналіз архітектурних підходів та принципів розробки, характерних для таких систем;
- вибір засобів програмної реалізації;
- проектування та розробка системи.

Об'єкт дослідження. Процес планування навчального навантаження кафедри

Предмет дослідження. Моделі, алгоритми та інформаційні технології планування навчального навантаження кафедри.

Методи досліджень: аналіз, синтез.

Апробація результатів дисертації. Результати дисертації було представлено на XV науково-технічній конференції «Сучасні інфокомунікаційні засоби».

Ключові слова. Інформаційна система, навантаження кафедри, планування навантаження, індивідуальний план, робочий навчальний план, автоматизація.

ABSTRACT

The master's dissertation on the topic "Modeling and automation of the planning of teaching load of the department" was completed by the student of Department of Digital Technologies in Energy of the Nuclear and Heat Power Institute Osypov Vladyslav from the 122 specialty "Computer Science" under the educational program "Digital Technologies in Energy" and consists of: introduction, 5 chapters ("Setting the task of automating the teaching load of the department", "Analysis of the subject area", "Description of the software implementation", "User's work with the system", "Development of a startup project"), conclusions to each chapter, general conclusions, a list of used sources, which includes 35 sources. The total volume of work is 116 pages.

Actuality of topic. Planning the teaching load of the department is a very important process in the work of the department and strongly affects the productivity of both students studying and teaching of disciplines by scientific and pedagogical staff. Although there are systems for automating this type of activity, none of them fully satisfy needs of users. This was a reason for the decision to conduct a study on this topic.

The purpose of the research is automating the planning of the teaching load of the department.

Tasks of the study:

- analysis of the process of planning the teaching load of the department;
- research of existing software solutions for automating the planning of the teaching load of the department;
- formation of requirements for the future system;
- analysis of architectural approaches and development principles characteristics of such systems;
- selection of software implementation tools;
- system design and development.

Object of study. The process of planning the teaching load of the department.

S**ubject of study.** Models, algorithms and information technologies for planning the teaching load of the department.

Research methods: analysis, synthesis.

Approbation of the results of the dissertation. The results of the dissertation were presented at the XV Scientific and Technical Conference “Modern Information Communication Tools”.

Keywords. Information system, department teaching load, teaching load planning, individual plan, work study plan, automation.

ЗМІСТ

ВСТУП

Розвиток технологій у сучасному світі зайшов так далеко, що важко і уявити життя пересічної людини без допомоги різних пристроїв та технологічних засобів. Вони значно спрощують не тільки повсякденне життя, а і робочі процеси. «Комп'ютеризація» дійшла до абсолютно всіх сфер діяльності, якими може займатись людина. Не обійшов цей процес і освітні заклади, де протягом останніх років застосовується все більше технологічних рішень для спрощення як процесу навчання студентів, так і робочих процесів співробітників навчальних закладів.

Сучасна школа або університет просто не може існувати, якщо не будуть застосовувати новітні інформаційні системи та програмні рішення, бо це спричинить значне відставання у освітніх процесах. Без таких програм складно функціонувати усім підрозділам навчальних закладів: бухгалтерії, деканатам тощо.

Для комфортної та ефективної роботи науково-педагогічного працівника необхідною умовою є правильно сформований розклад навчання. Передує цьому процесу планування навчального навантаження кафедри.

Цей вид діяльності методистів є надзвичайно важливим, і в той самий час громіздким та рутинним. Працівник, відповідальний за цей процес, має бути зосередженим та уважним, так як дуже легко припуститися помилки під час роботи з великою кількістю обчислень. Також постійно потрібно тримати біля себе інформацію, скільки у кожного викладача вже є годин навантаження, щоб триматись у межах мінімальних та максимальних обмежень, зважаючи ще на види дільностей, які призначені для викладачів з певними вченими ступенями. Відсутність програмного забезпечення, що дозволило б автоматизувати даний процес або хоча б частково оптимізувати час роботи з ним стала причиною проведення дослідження та розробки програмного забезпечення, що виконувало б хоча б частину задач, що стоять перед методистами.

Метою дослідження є з'ясування рівня можливої автоматизації планування навчального навантаження кафедри, проектування структури та розробка системи, визначення її потенційних можливостей та їх реалізація.

Для досягнення поставленої мети було сформульовано наступні завдання до дослідження:

- аналіз процесу планування навчального навантаження кафедри;
- дослідження існуючих програмних рішень для автоматизації планування навчального навантаження кафедри;
- аналіз архітектурних підходів та принципів розробки, характерних для таких систем;
- проектування та розробка системи.

Об’єкт проведеного дослідження являє собою планування навчального навантаження кафедри, в той час як предметом дослідження являється автоматизація планування навчального навантаження кафедри.

За час розробки системи було удосконалено існуючі програмні підходи до процесу планування навчального навантаження кафедри та вперше створено систему, що відповідає повністю відповідає вимогам, поставленим методистами.

Під час розробки основними методами дослідження, що використовувалися, були аналіз і синтез. Під час фази аналізу процес планування навчального навантаження кафедри розкладався на етапи, кожен з яких розглядався, як окрема задача. Під час фази синтезу реалізовувалося програмне забезпечення, що здатне вирішити кожну з задач.

Розроблена система автоматизації планування навчального навантаження кафедри дозволяє автоматизувати всі основні складові даного процесу. Програмне забезпечення може застосовуватися методистами для оптимізації часу, що витрачається на процес планування.

Основні положення роботи доповідались на XV науково-технічній конференції «Сучасні інфокомунікаційні технології».

Робота складається зі вступу, 5 розділів, висновків до розділів, загальних висновків, списку використаних джерел (35 джерел, у тому числі 27 – іноземною мовою), додатків на 5 сторінках. Загальний обсяг дисертації - 116 сторінок. Основний зміст викладено на 96 сторінках. Роботу проілюстровано 24 рисунками, 21 таблицею.

1 ПОСТАНОВКА ЗАДАЧІ АВТОМАТИЗАЦІЇ НАВЧАЛЬНОГО НАВАНТАЖЕННЯ КАФЕДРИ

Розвиток технологій, що переживає сучасний світ, значно полегшує життя людства у всіх сферах. Та реальність показує, що за покращенням слідує неперервне підвищення вимог, яке пов’язане з інтеграцією технологій у різні сфери. Різні компанії змушені впроваджувати інноваційні підходи автоматизації праці, щоб мати змогу залишатися конкурентоспроможними і не витрачати час працівників на рутинну роботу. Ця проблема дійшла і до навчальних закладів, де велика кількість роботи, пов’язаної з документообігом, досі виконується виключно працівниками, з мінімальною кількістю автоматизації процесів, яка найчастіше створена силами самих працівників. Однією з таких громіздких задач є формування навчального навантаження кафедри. Цей процес вимагає від відповідального за нього виключної уваги та зосередженості. Так як цей процес є доволі рутинним і пов’язаний в більшості з обчисленням певних величин, це значно підвищує шанси помилки у обчисленнях через одноманітність роботи та важкість знаходження помилки через велику кількість розрахунків. Задля автоматизації цього процесу було проведено дане дослідження та розроблено програмний продукт.

У даному розділі буде розглянуто мету дослідження можливості автоматизації планування навчального навантаження кафедри, розглянуто вимоги до майбутнього програмного забезпечення та задач, які воно має вирішувати, а також вхідні та вихідні дані роботи майбутньої системи.

1.1 Призначення програмного забезпечення

Головним призначенням створеного програмного продукту є автоматизація процесу формування навчального навантаження кафедри. Кінцевою точкою цієї задачі є створення форми К-4, що містить навантаження кожного викладача.

Потенційними користувачами створеної системи є працівники кафедри (факультету), що займаються розподіленням навантаження на викладачів.

1.2 Задачі, що вирішує ПЗ

Основними задачами, що вирішуються створеним програмним забезпеченням, є:

- аналіз вхідного файлу з робочим навчальним планом та вибір дисциплін, що викладаються на даній кафедрі;
- формування форми К-3;
- закріплення певної дисципліни (або певних видів роботи) за викладачем;
- перевірка навантаження викладача на відповідність встановленим нормам;
- введення додаткової інформації стосовно кількості потоків ті підгруп у групі;
- введення даних стосовно позааудиторної діяльності викладача (керівництва дипломними роботами, консультацій та ін.);
- можливість порівняння навантаження з минулими, сформованими у даній системі;
- можливість вносити зміни до БД викладачів;
- формування форми К-4;
- формування файлу з навантаженням для окремого викладача.

1.3 Вхідні дані

На початковому етапі роботи з системою вхідним файлом є робочий навчальний план, приклад якого зображений на рисунку 1.1. Робочі навчальні плани складаються на основі відповідних навчальних планів на кожен рік підготовки за відповідною освітньою програмою, що мають на меті вдосконалення

змісту навчання, конкретизації планування навчального процесу і закріплення освітніх компонентів (навчальних дисциплін) за кафедрами на наступний навчальний рік[1]. Робочі навчальні плани та навчальні плани складаються робочими групами випускових кафедр з можливим залучення працівників інших кафедр. Керує процесом роботи групи гарант відповідної освітньої програми, а контролює виконання вимог до навчальних і робочих навчальних планів декан факультету або директор інституту. Вимоги до оформлення, змісту та структури навчальних і робочих навчальних планів затверджуються наказом ректора про підготовку до нового навчального року в НТУУ «КПІ ім. Ігоря Сікорського». Затверджуються робочі навчальні плани проректором з навчальної роботи вищого навчального закладу.

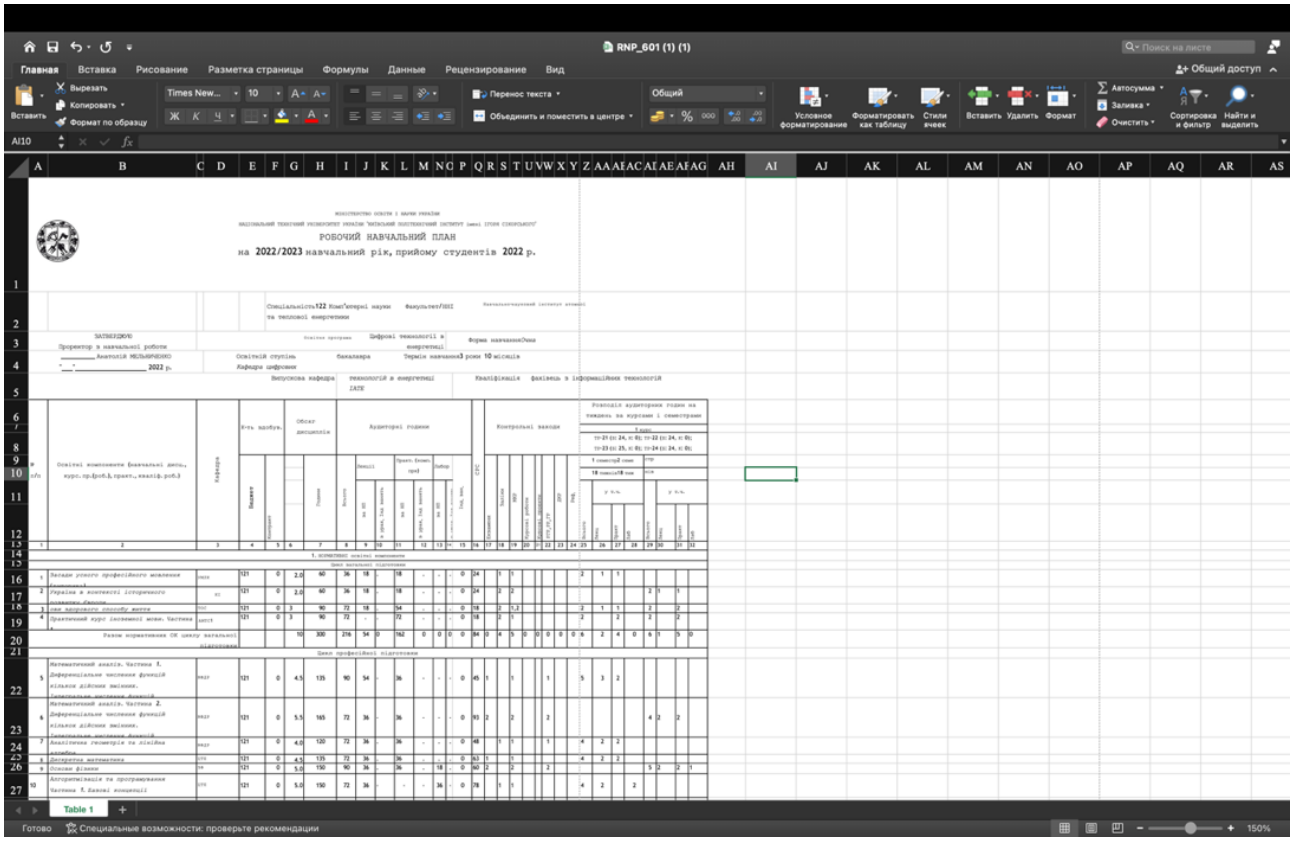

Рисунок 1.1 – Робочий навчальний план

Для формування форми К-3 необхідно додатково ввести інформацію щодо кількості потоків на курсі та кількості підгруп у групах, де кількість студентів перевищує 15 студентів.

Щоб сформувати форму К-4, користувач повинен ввести інформацію про індивідуальні заняття викладача, керівництво практиками, керівництво атестаційними роботами, консультування по випускному екзамену, рецензування атестаційних робіт (бакалаври, магістри ОПП та ОНП), роботу в ЕК, вступні екзамени керівництво (аспіранти, стажери), консультування докторантів.

1.4 Вихідні дані

Перший етап роботи з системою формує форму К-3, приклад якої представлений на рисунку 1.2.

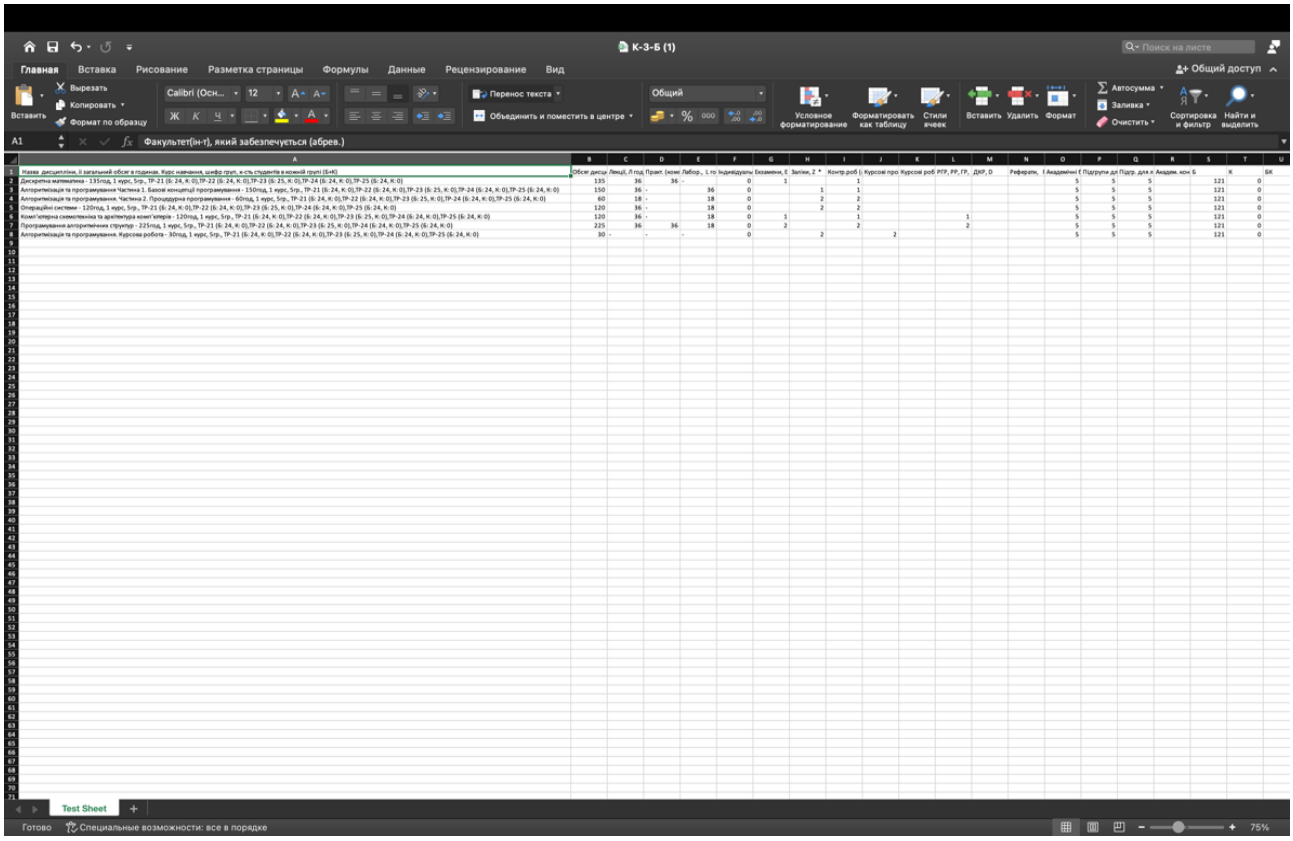

Рисунок 1.2 – Форма К-3

Форма К-3 містить інформацію про назви дисциплін, їхній обсяг у годинах, групи, що її вивчають, кількість годин, виділених для аудиторних занять (лекцій,

практичних та лабораторних занять), індивідуальних занять, контрольних заходів (екзаменів, заліків, контрольних робіт (модульних, тематичних), курсових робіт та проєктів, РГР, РР, ГР, ДКР), рефератів, консультацій, кількість академічних груп, підгруп для практичних та лабораторних занять, потоків та загальну кількість годин.

Система також має надати можливість користувачеві отримати файл з формою К-4, приклад якої наведено на рисунку 1.3.

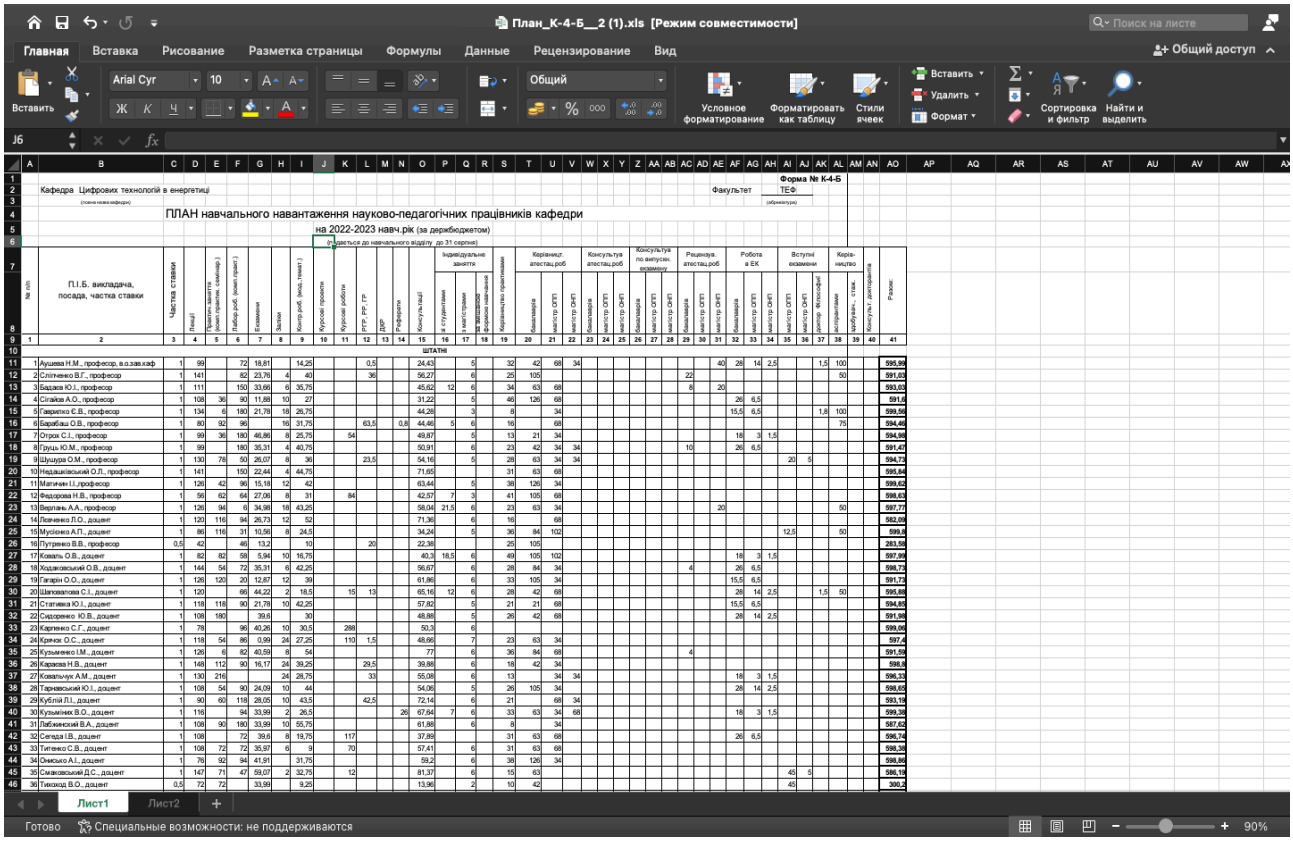

Рисунок 1.3 – Форма К-4

Форма К-4 є кінцевим файлом в роботі з навчальним навантаженням кафедри. Файл містить інформацію про ім’я, посаду, частку ставки викладача, обсяг його аудиторної та позааудиторної діяльності. До аудиторних видів діяльності входять лекції, практичні заняття (семінари), лабораторні роботи (комп’ютерні практикуми), екзамени, контрольні роботи (модульні, тематичні), курсові проєкти, курсові роботи, розрахунково-графічні роботи (розрахункові роботи, графічні роботи), домашні контрольні роботи, реферати, консультації. До

позааудиторної роботи належать індивідуальні заняття (зі студентами, магістрами), керівництво практиками, керівництво атестаційними роботами (бакалаврів, магістрів ОПП та ОНП), консультування атестаційних робіт (бакалаврів, магістрів ОПП та ОНП,) рецензування атестаційних робіт (бакалаврів, магістрів ОПП та ОНП), робота в Державній екзаменаційній комісії (бакалаврів, магістрів ОПП та ОНП), керівництво (аспірантами та стажерами), заняття з аспірантами та консультування докторантів.

Система також має надати можливість користувачеві завантажити окремий файл з індивідуальним планом навчального навантаженням для окремого викладача. Приклад файлу з індивідуальним навчальним навантаженням представлено на рисунку 1.4.

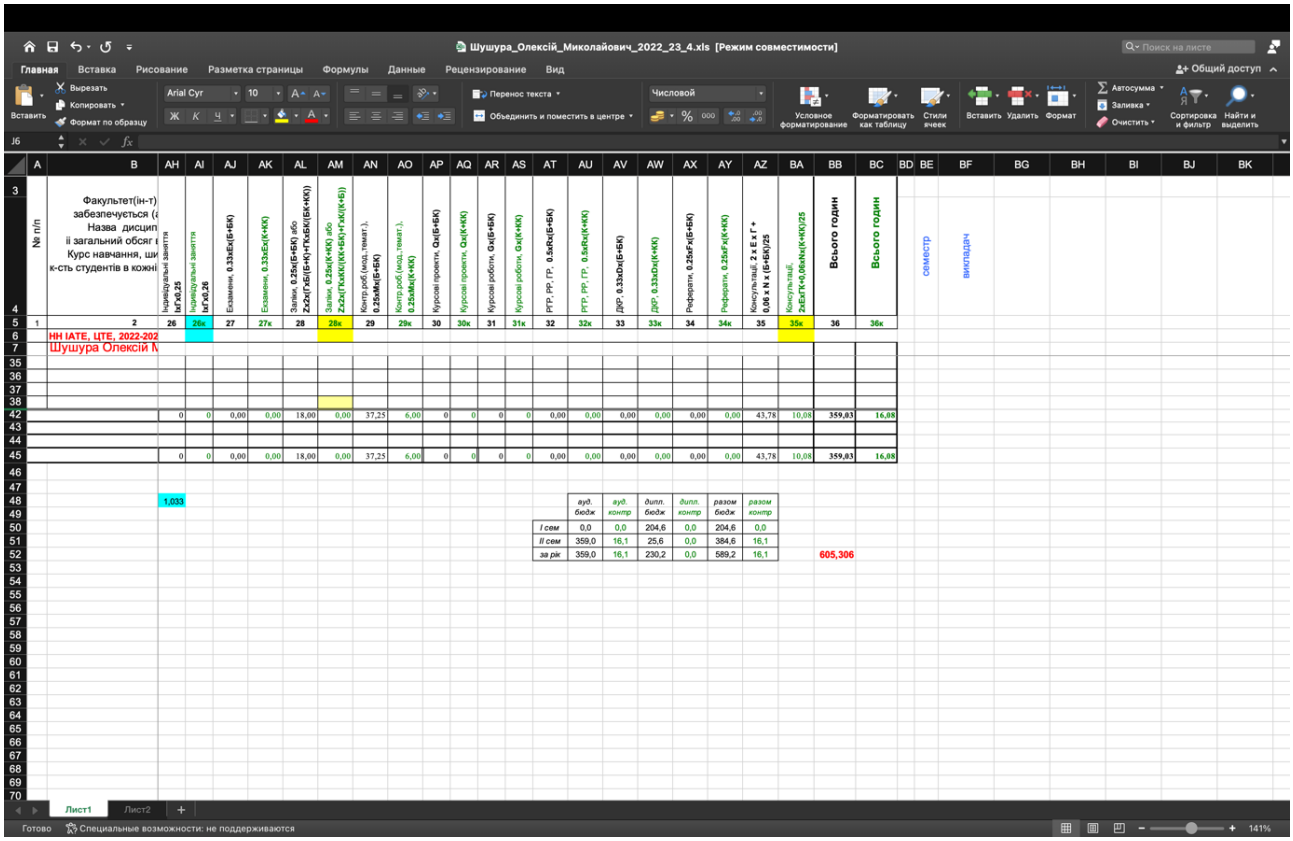

Рисунок 1.4 – Файл навантаження окремого викладача

Усі вихідні дані, надані системою, користувач здатен завантажити з неї.

1.5 Висновки до розділу 1

У цьому розділі визначено мету дослідження, сформовано задачі, які має вирішувати система, створена на основі результатів дослідження, розглянуто наявні вхідні файли роботи з майбутнім програмним забезпеченням та вихідні файли, які користувач має отримати в результаті користування системою. До основних задач, які має реалізувати система входять:

- надати можливість сформувати файл форми К-3;
- надати можливість створити індивідуальний план навантаження;
- надати можливість сформувати план навчального навантаження науково-педагогічних працівників кафедри.

2 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Після визначення мети дослідження, поверхневого аналізу вхідних та вихідних даних, консультацій з методистами та формулювання основних задач, які дослідження та майбутня система мають вирішувати, стало зрозуміло, що задача автоматизації планування навчального навантаження кафедри є більш складною, ніж вона здається на перший погляд. Цей процес складається з певної кількості підзадач, які вимагають від працівника, який їх виконує, виключною уважності та зосередженості. У цьому розділі буде розглянуто та проаналізовано існуючі програмні рішення, що частково або повністю виконують поставлену у дослідженні задачу, їх переваги та недоліки, детально проаналізовано вхідні документи та весь процес планування навчального навантаження кафедри від аналізу робочого навчального плану та формування плану навчального навантаження науково-педагогічних працівників кафедри.

2.1 Огляд існуючих програмних рішень

Попередніми дослідженнями в межах кафедри було виявлено відсутність програмних засобів, що вирішують задачу автоматизації процесу планування навчального навантаження кафедри. Результатом пошуку систем з підходящою функціональністю стали три системи управління навчальним процесом: АСУ «ВНЗ» та пакети програм «Політек-СОФТ».

Головними недоліками даних програмних рішень є:

- складний у розумінні користувацький інтерфейс;
- погано реалізована або відсутня кросплатформеність;
- ускладений супровід технології.

Альтернативним варіантом частково автоматизувати процес планування навчального навантаження кафедри та оптимізувати роботу методичного відділу є використання інструментів, які надаються пакетом Microsoft Office, а саме Microsoft Excel і Micorsoft Access.

Саме тому постало завдання створення комплексної, зрозумілої для користувача системи автоматизації планування навчального навантаження кафедри, яку буде легко підтримувати та впроваджувати у вищих навчальних закладах. Далі у розділі буде розглянуто наведені вище програмні засоби.

2.1.1 Пакети програм «Політек-СОФТ»

Співпраця ПП «Політек-СОФТ» з вищими навчальними закладами триває вже більше 20 років, що позитивно впливає на якість програмного забезпечення та підкреслює його актуальність. Системи вирішують задачі більшої частини процесів, що відбуваються у вищих навчальних закладах[2].

На сьогоднішній день програмне забезпечення «Політек-СОФТ» представлено наступними пакетами програм:

- «Колоквіум»;
- «Бібліограф»;
- «ПС-Персонал»;
- «Деканат»;
- «ПС-Абітурієнт» (до вересня 2022 року)[3].

Пакет програм «Колоквіум» надає користувачеві можливості автоматизувати процес тестування та опитування студентів. До основних можливостей даного пакету належать:

- наявність зручної форми створення тестів та опитувань;
- можливість проводити у різних режимах тестування студентів та оцінювати їх на основі цих тестів;
- можливість у реальному часі контролювати процес проведення тестування;
- можливість аналізу результатів тестування через доступ до бази даних з результатами;
- можливість створити звіт за результатами проведених тестів та опитувань.

До переваг також можна віднести сумісність з іншими продуктами «Політех-СОФТ», якісний та зрозумілий інтерфейс, широкий вибір режимів проведення тестів та їх оцінювання та досить велику базу даних.

Пакет програм «Бібліограф» надає користувачеві можливості автоматизації процесів, пов'язаних з діяльністю бібліотеки. Пакет надає можливості для формування даних про:

- бібліографічний опис книг, статей;
- електронний варіант книги сумарного обліку;
- анкетні дані читачів бібліотеки;
- електронний квиток читача;
- обіг видань;
- забезпеченість літературою;
- статистику видачі літератури;
- статистику обліку читачів.

Пакети «ПС-Персонал» та «Деканат» містять функціонал, що в певній мірі вирішує завдання дослідження. Перший призначений для автоматизації процесів відділу кадрів, другий – планування навчального процесу. Пакет «ПС-Персонал» надає користувачеві можливість формування даних про:

- освіту робітника;
- сімейний стан;
- проходження атестацій та підвищення кваліфікацій;
- службові переміщення;
- наказів про переміщення;
- військовий облік;
- відпустки;
- подяки та догани.

Пакет «Деканат», в свою чергу, дозволяє формувати дані про:

- структуру навчального процесу;
- викладачів, їх планове навантаження та розклад;
- фактичну роботу викладача по дисципліні;
- успішність студентів;

- наявність корпусів та аудиторій.

Саме цей пакет найбільше відповідає поставленій у дослідженні задачі. Проте головним недоліком даного програмного забезпечення є відсутність підтримки кредитно-модульної системи, яка запроваджена у більшості навчальних закладів України.

2.1.2 Автоматична система управління «ВНЗ»

Система АСУ «ВНЗ» є сучасним автоматизованим інструментом для управління навчальним процесом вищих навчальних закладів. На даний момент система є частиною системи загальнодержавного рівня ІВС «ОСВІТА». Дане програмне забезпечення представлено трьома компонентами: «Студмістечко», «Приймальна комісія» та «Деканат»[4].

Автоматизована система «Студмістечко» використовується для автоматизації процесів, пов’язаних з діяльністю студмістечка. Система дозволяє:

- поселяти студентів;
- спростити документообіг;
- керувати пропускною системою;
- спростити друк документації, заснованої на внесених даних;
- контролювати житловий фонд, сплату проживання тощо.

Автоматизована система «Приймальна комісія» надає можливості для автоматизації процесів, пов’язаних з діяльністю приймальної комісій. Дане програмне забезпечення значно оптимізує часові витрати співробітників приймальних комісій та надає достовірні та актуальні дані. До основних можливостей системи відносяться:

- можливість опрацювати дані актуального стану прийому документів;
- можливість ввести електронні картки для абітурієнтів;
- можливість синхронізувати дані з ресурсами ЄДБО;
- можливість створювати широкий спектр звітів;
- можливість сформувати конкурсні списки;

- можливість аналізувати дані про рейтинг абітурієнтів, відслідковувати недостовірні дані;
- можливість пошуку по системі використовуючи обмежену кількість інформації;
- можливість вносити бали з шкільних атестатів, дипломів, сертифікатів УЦОЯО;
- можливість надсилати дані для замовлення студентських квитків.

Компонентом, що найповніше задовільняє ціль дослідження, є автоматизована система «Деканат», яка використовується для оптимізації роботи методичних відділів шляхом автоматизації значної кількості процесів та зменшення обсягу паперового документообігу[5]. На сьогоднішній день система складається з 13 модулів:

- відділ кадрів;
- накази;
- журнал відомостей;
- студенти;
- навчальні плани;
- журнал;
- розклад;
- переклад на наступний навчальний рік;
- сесії;
- контракти;
- індивідуальний план викладача;
- результати сесії;
- навантаження кафедр.

Ця система надає широкий спектр можливостей для:

- створення робочих навчальних планів на основі навчальних планів;
- створення індивідуальних навчальних планів викладачів;
- закріплення викладачів для формування навантаження за робочими навчальними планами;
- розробки навчальних та робочих навчальних планів на рік;

- формування розкладу.

АСУ «ВНЗ» реалізована засобами Delphi та баз даних SQL. Головним недоліком даної системи є той факт, що не передбачено подальшого оновлення систем, тобто навчальний заклад буде змушений знову купляти системи та імпортувати в неї дані. Налаштування та супровід стабільної роботи системи також здійснюється клієнтською стороною.

2.1.3 Програма Microsoft Excel

Microsoft Excel – програма для роботи з електронними таблицями, створена компанією Microsoft та адаптована під більшість популярних операційних систем[6]. Приклад інтерфейсу Microsoft Excel зображено на рисунку 2.1.

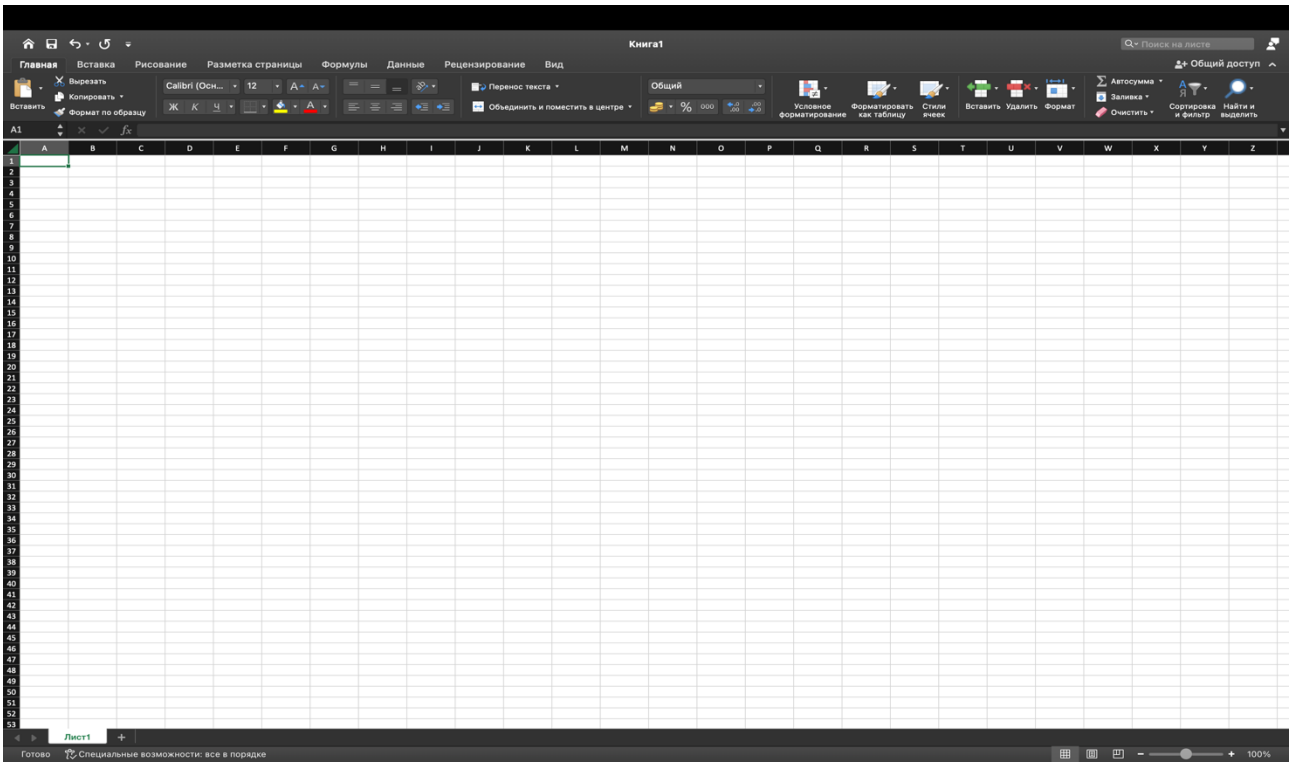

Рисунок 2.1 – Інтерфейс користувача Microsoft Excel

Являє собою надзвичайно потужний інструмент, що надає надзвичайно велику варіативність операцій для обробки табличних даних.

До інших важливих можливостей Microsoft Excel можна віднести створення графіків, різних видів діаграм. Та найважливішим інструментом, який надає програма є можливість використовувати вбудовані функції[7]. Функції є фактично спрощеним варіантом програмування: вони надають умовні оператори, оператори для різних математичних дій тощо.

Не менш важливим інструментом є макроси. Макроси можна писати використовуючи мову програмування Visual Basic. Завдяки цьому інструменту можна частково автоматизувати частину роботи Microsoft Excel.

Саме ці можливості спонукають деяких методистів використовувати дане програмне забезпечення для планування навчального навантаження кафедри. Головним недоліком даного підходу до автоматизації є необхідність методиста мати досить вичерпні знання у роботі Microsoft Excel та хоча б мінімальні навички програмування для написання макросів. У технічних вищих навчальних закладах це рідко є проблемою, але для гуманітарних ВНЗ може стати нездоланною перешкодою.

2.1.4 Програма Microsoft Access

Microsoft Access – система для керування базами даних, створена компанією Microsoft, що поєднує рушій для керування базами даних Access (англ. ACE, скорочено Access Database Engine) з графічним інтерфейсом користувача та інструментами розробки. Користувачеві також надається можливість імпортувати або приєднуватись до інших програм або баз даних[9]. Як і Microsoft Excel та і нші програми пакету Microsoft Office, Access. Підтримує розробку мовою Visual Basic. Дане програмне забезпечення може використовуватися як варіант фронтенду, в той час як інші програми будуть діяти як бекенд таблиця, як наприклад Microsoft SQL Server, або продукти інших команій, як Oracle. Також такі технології, як Visual Basic, ASP.NET або Visual Studio .NET використовують формат Access баз даних для своїх таблиць. Microsoft Access також можна легко зробити частиною більш складного комплексу, куди можна інтегрувати інші технології, як Microsoft Excel,

Microsoft Word тощо. Приклад інтерфейсу користувача представлено на рисунку 2.2.

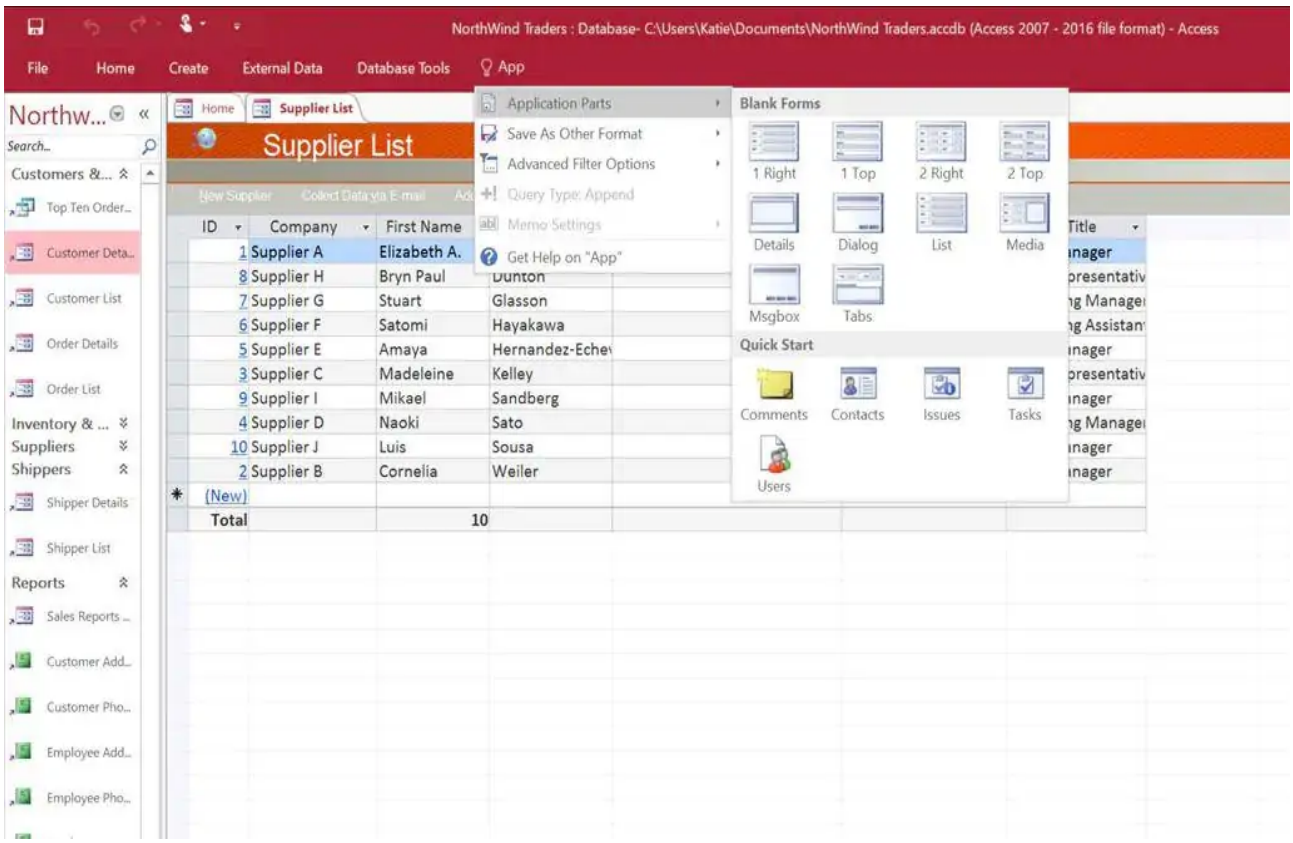

Рисунок 2.2 – Інтерфейс користувача Microsoft Access

Хоча дане програмне забезпечення і надає велику кількість можливостей користувачеві, і при належному використанні може бути застосовано для автоматизації певних процесів планування навчального навантаження кафедри, воно абсолютно не підходить для роботи у гуманітарних ВНЗ. Якщо для роботи Microsoft Excel користувачеві достатньо розуміти, як застосовувати формули, то щоб працювати з Microsoft Access, методист має мати достатній бекграунд, як розробник, для розуміння структури та методів роботи з базами даних. Тому дане програмне забезпечення підходить виключно для вищих навчальних закладів, де працівники методичного відділу, які часто є викладачами з кафедри, хоча б віддалено розуміють програмування.

2.2 Загальний огляд етапів процесу формування навчального навантаження

Процес формування начального навантаження кафедри починається з аналізу робочого навчального плану курсу. Робочий навчальний план є частиною навчального плану і формується на основі аналізу сукупності індивідуальних навчальних планів здобувачів вищої освіти на навчальний рік. Цей документ містить інформацію про:

- навчальний заклад;
- навчальний рік, на який він був складений;
- освітній ступінь;
- спеціальність (код і назва);
- спеціалізацію (освітньо-професійну програму);
- випускову кафедру;
- факультет (ННІ);
- форму навчання;
- термін навчання;
- кваліфікацію;
- нормативні освітні компоненти (цикли загальної і професійної підготовки);
- вибіркові освітні компоненти;
- курс;
- шифри груп;
- кількість студентів у кожній групі.

Кожен рядок, що містить нормативний або вибірковий освітній компонент, в свою чергу включає в себе дані про:

- назву навчальної дисципліни (кваліфікаційної роботи, курсової роботи або проєкту);
- кафедру, що викладає даний компонент;
- кількість здобувачів (бюджетних і контрактних);
- обсяг дисципліни (у годинах та кредитах ECTS);

- кількість аудиторних годин за навчальним планом та з урахуванням Індивідуальних занять (для лекцій, практикумів та лабораторних занять), індивідуальних занять;
- самостійну роботу студента (СРС);
- кількість годин на контрольні заходи (екзамени, заліки, МКР, курсові роботи, курсові проєкти, РГР, РР, ГР, ДКР, реферати);
- розподіл аудиторних годин на тиждень за курсами і семестрами.

Приклад структури освітнього компоненту зображено на рисунку 2.3.

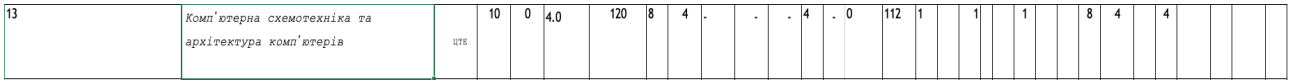

Рисунок 2.3 – Приклад структури освітнього компоненту у робочому навчальному плані

Під час роботи з робочим навчальним планом методисту потрібно вибрати тільки ті освітні компоненти, що викладаються його кафедрою, після чого він починає формувати файл форми К-3. Форма К-3 формується у вигляді таблиці окремо для студентів різної форми навчання (денна, заочна, за кошти фізичних осіб, за державні кошти). Цей файл містить перелік дисциплін та всю інформацію про них. Кожний рядок файлу містить інформацію про:

- назву дисципліни;
- загальний обсяг у годинах;
- кількість лекцій у годинах;
- кількість практичних занять (комп'ютерних практикумів, семінарів) у годинах;
- кількість лабораторних занять у годинах;
- кількість індивідуальних занять у годинах;
- екзамени;
- заліки;
- контрольні роботи (модульні, тематичні);
- курсові роботи;

- курсові проєкти;
- РГР, РР, ГР;
- ДКР;
- реферати;
- кількість академічних бюджетних груп;
- кількість підгруп для практичних занять;
- кількість підгруп для лабораторних занять;
- кількість академічних контрактних груп;
- загальну кількість студентів бюджетної форми навчання у бюджетних групах;
- загальну кількість студентів контрактної форми навчання у бюджетних групах;
- загальну кількість студентів бюджетної форми навчання у контрактних групах;
- загальну кількість студентів контрактної форми навчання у контрактних групах;
- кількість потоків;
- обчислену загальну кількість лекцій у годинах;
- обчислену загальну кількість практичних занять у годинах;
- обчислену загальну кількість лабораторних занять у годинах;
- обчислену загальну кількість індивідуальних занять у годинах;
- обчислену кількість екзаменів у годинах;
- обчислену кількість заліків у годинах;
- обчислену кількість контрольних робіт у годинах;
- обчислену кількість курсових проєктів у годинах;
- обчислену кількість курсових робіт у годинах;
- обчислену кількість РГР, РР, ГР у годинах;
- обчислену кількість ДКР у годинах;
- обчислену кількість рефератів у годинах;
- обчислену кількість консультацій у годинах;
- загальну кількість годин на дисципліну.

Виділивши необхідну інформацію з робочого навчального плану та заповнивши частину форми К-3, необхідно почати роботу над обчисленням загальних кількостей годин, виділених на ті чи інші види діяльностей. Навчальне навантаження розраховується, виходячи з астрономічної години або 60 хвилин. При розрахунках, пов'язаних з аудиторною роботою, академічна година або 45 хвилин зараховується, як астрономічна.

Починаючи саме з цього етапу починається та частина роботи методиста, що вимагає зосередженості та уважності через той факт, що у великій кількості обчислень легко припуститись помилки, яку досить важко буде відслідкувати у майбутньому. Для обчислення величин використовуються формули, складені на основі затверджених на державному рівні величин[9].

Для обчислення загальної кількості годин, виділених на лекційні заняття з певної дисципліни використовується формула вигляду:

$$Л_{заг} = Л * Р,$$

де $Л_{заг}$ – загальна кількість лекційних годин з дисципліни,

Л – кількість планових лекційних годин,

Р – кількість академічних потоків

З цієї формули можна зрозуміти, що загальна кількість годин, що витрачається на читання лекцій з дисципліни дорівнює кількості планових годин помноженій на кількість потоків, так як лекції з предметів зазвичай читаються для всього потоку, який вивчає дану дисципліну, а не окремих груп.

Щоб обчислити загальну кількість годин для практичних занять, потрібно використати формулу, що має наступний вигляд:

$$П_{заг} = П * ГП,$$

де $П_{заг}$ – загальна кількість годин для практичних занять з дисципліни,

П – кількість планових годин для практичних занять,

ГП – сумарна кількість підгруп для практичних занять

За формулою загальна кількість годин для практичних занять дорівнює добутку планової кількості годин та сумарній кількості підгруп для практичних занять у всіх групах, у яких читається дана дисципліна[10]. Цей спосіб обчислення зумовлений тим, що кількість підгруп у групах може відрізнятися у зв'язку з

кількістю студентів у групах або розподілом різних видів робіт з дисциплін між викладачами.

Для знаходження загальної кількості годин для лабораторних занять виведено наступну формулу:

$$L_{заг} = L * ГL,$$

де $L_{заг}$ – загальна кількість годин для лабораторних занять з дисципліни,

L – кількість планових годин для лабораторних занять,

ГL – сумарна кількість підгруп для лабораторних занять

Формула дає зрозуміти, що загальна кількість годин для лабораторних занять знаходиться шляхом множення кількості планових годин на сумарну кількість підгруп для лабораторних занять у всіх групах, що вивчають дисципліну. Причиною цього є той факт, що через різну кількість студентів у групах та різний ступінь завантаженості викладачів, кількість підгруп у різних групах може відрізнятися.

Для обчислення загальної кількості годин, необхідних для проведення індивідуальних занять, використовується формула наступного вигляду:

$$I_{заг} = I * Г * 0{,}25,$$

де $I_{заг}$ – загальна кількість годин для індивідуальних занять з дисципліни,

I – кількість планових годин для індивідуальних занять,

Г – загальна кількість груп, для яких читається дисципліна

З формули можна зробити висновок, що для знаходження загальної кількості годин для індивідуальних занять необхідно знайти добуток планових годин для індивідуальних занять і загальної кількості груп, яким читається дисципліна, та помножити його на коефіцієнт 0,25. Це пояснюється тим, що на кожну групу, на яку заплановано індивідуальні заняття, виділяється четверть години або 25% від години.

Загальна кількість годин, виділених на проведення екзаменів, обчислюється за формулою:

$$E_{заг} = E * (Б + БК) * 0{,}33,$$

де $E_{заг}$ – загальна кількість годин, необхідна для проведення екзаменів,

E – планова кількість годин для проведення екзаменів,

Б – кількість студентів на бюджетній формі навчання у бюджетних групах,

БК – кількість студентів на бюджетній формі навчання у контрактних групах

Інший варіант формули для обчислення даної величини має наступний вигляд:

$$Е_{заг} = Е * (К + КК) * 0{,}33,$$

де $Е_{заг}$ – загальна кількість годин, необхідна для проведення екзаменів,

Е – планова кількість годин для проведення екзаменів,

К – кількість студентів на контрактній формі навчання у бюджетних групах,

КК – кількість студентів на контрактній формі навчання у контрактних групах

Формули використовуються залежно від виду форми К-3, яку формує методист – для бюджетної чи контрактної форми навчання відповідно. З формули можна зрозуміти, що загальна кількість годин на проведення екзаменів з окремої дисципліни дорівнює добутку планової кількості екзаменаційних годин, загальної суми кількості студентів бюджетної форми навчання у бюджетних та контрактних групах або загальної суми кількості студентів контрактної форми навчання у бюджетних та контрактних групах та коефіцієнту 0,33. Це пояснюється тим, що час, який виділяється на прийом екзамену у одного студента, дорівнює 33% від години.

Формула для знаходження загальної кількості годин для проведення заліків з дисципліни виглядає наступним чином:

$$Z_{заг} = Z * Г * 2,$$

Де $Z_{заг}$ – загальна кількість годин, необхідна для проведення заліків,

Z – планова кількість годин для проведення заліків

Проаналізувавши формулу можна зробити висновок, що для знаходження загальної кількості годин на проведення заліків з дисципліни знаходиться шляхом обчислення добутку планової кількості годин для проведення заліків, загальної кількості груп, які вивчають дану дисципліни та коефіцієнту, що дорівнює 2. Ця величина обчислюється саме в такий спосіб у зв’язку з тим, що тривалість заліку у кожної групи дорівнює 2 годинам.

Для обчислення загальної кількості годин для проведення контрольних робіт (модульних, тематичних) використовується наступна формула:

$$М_{заг} = М * (Б + БК) * 0,25,$$

де $М_{заг}$ – загальна кількість годин, необхідна для проведення контрольних робіт,

М – планова кількість годин для проведення контрольних робіт,

Б – кількість студентів бюджетної форми навчання у бюджетних групах,

БК – кількість студентів бюджетної форми навчання у контрактних групах

Інший варіант формули для обчислення даної величини виглядає наступним чином:

$$М_{заг} = М * (К + КК) * 0,25,$$

де $М_{заг}$ – загальна кількість годин, необхідна для проведення контрольних робіт,

М – планова кількість годин для проведення контрольних робіт,

К – кількість студентів контрактної форми навчання у бюджетних групах,

КК – кількість студентів контрактної форми навчання у бюджетних групах

Після огляду формули можна зробити висновок, що загальна кількість годин для проведення контрольних робіт знаходиться шляхом обчислення добутку планової кількості годин для проведення контрольних робіт, загальної суми студентів бюджетної форми навчання у бюджетних та контрактних групах або студентів контрактної форми навчання у бюджетних та контрактних групах та коефіцієнту 0,25. Даний спосіб обчислення зумовлений тим, що час, який виділяється на проведення контрольної роботи для одного студента, дорівнює 25% від години.

Щоб знайти загальну необхідну кількість годин для роботи над курсовими проєктами, необхідно використати формулу:

$$Q_{заг} = Q * (Б + БК),$$

де $Q_{заг}$ – загальна кількість годин для роботи над курсовими проєктами,

Q – планова кількість годин для роботи над курсовими проєктами,

Б – кількість студентів бюджетної форми навчання у бюджетних групах,

БК – кількість студентів бюджетної форми навчання у контрактних групах

Альтернативний варіант запису формули для даної величини виглядає наступним чином:

$$Q_{заг} = Q * (К + КК),$$

де $Q_{заг}$ – загальна кількість годин для роботи над курсовими проєктами,

Q – планова кількість годин для роботи над курсовими проєктами,

К – кількість студентів контрактної форми навчання у бюджетних групах,

КК – кількість студентів контрактної форми навчання у контрактних групах

З вигляду формули можна зрозуміти – щоб обчислити загальну кількість годин для роботи над курсовими проєктами, необхідно знайти добуток планової кількості годин для роботи над курсовими проєктами та суми студентів бюджетної форми навчання у бюджетних та контрактних групах або суми студентів контрактної форми навчання у бюджетних та контрактних групах.

Для знаходження кількості годин, необхідних для роботи над курсовими роботами, виведено формулу:

$$G_{заг} = G * (Б + БК),$$

де $G_{заг}$ – загальна кількість годин для роботи над курсовими роботами,

G – планова кількість годин для роботи над курсовими роботами,

Б – кількість студентів бюджетної форми навчання у бюджетних групах,

БК – кількість студентів бюджетної форми навчання у контрактних групах

Інший варіант формули для знаходження цієї величини має наступний вигляд:

$$G_{заг} = G * (К + КК),$$

де $G_{заг}$ – загальна кількість годин для роботи над курсовими роботами,

G – планова кількість годин для роботи над курсовими роботами,

Б – кількість студентів бюджетної форми навчання у бюджетних групах,

БК – кількість студентів бюджетної форми навчання у контрактних групах

Запис формули дає зрозуміти, що загальна кількість годин для роботи над курсовими роботами обчислюється шляхом знаходження добутку планової кількості годин для роботи над курсовими роботами та суми кількості студентів бюджетної форми навчання у бюджетних та контрактних групах або контрактної форми навчання у бюджетних та контрактних групах.

Для обчислення кількості годин для розрахунково-графічних, розрахункових та графічних робіт застосовується формула:

$$R_{заг} = R * (Б + БК) * 0{,}5,$$

де $R_{заг}$ – загальна кількість годин для РГР, РР, ГР,

R – планова кількість годин для РГР, РР, ГР,

Б – кількість студентів бюджетної форми навчання у бюджетних групах,

БК – кількість студентів бюджетної форми навчання у контрактних групах

Альтернативним варіантом формули для знаходження даної величини є такий запис:

$$R_{заг} = R * (К + КК) * 0{,}5,$$

де $R_{заг}$ – загальна кількість годин для РГР, РР, ГР,

R – планова кількість годин для РГР, РР, ГР,

Б – кількість студентів бюджетної форми навчання у бюджетних групах,

БК – кількість студентів бюджетної форми навчання у контрактних групах

Запис формули демонструє, що для обчислення загальної кількості годин для РГР, необхідно планову кількість годин для РГР помножити на суму кількості студентів бюджетної форми навчання у бюджетних та контрактних групах або контрактної форми у бюджетних та контрактних групах та на коефіцієнт 0,5. Таке формулювання пояснюється тим, що на розрахункову роботу одного студента виділяється 50% від години.

Для того, щоб знайти загальну кількість годин для домашніх контрольних робіт, було виведено формулу:

$$D_{заг} = D * (Б + БК) * 0{,}33,$$

де $D_{заг}$ – загальна кількість годин для ДКР,

D – планова кількість годин для ДКР,

Б – кількість студентів бюджетної форми навчання у бюджетних групах,

БК – кількість студентів бюджетної форми навчання у контрактних групах

Також випадки, коли використовується видозмінений варіант запису даної формули:

$$D_{заг} = D * (К + КК) * 0{,}33,$$

де $D_{заг}$ – загальна кількість годин для ДКР,

D – планова кількість годин для ДКР,

Б – кількість студентів бюджетної форми навчання у бюджетних групах,

БК – кількість студентів бюджетної форми навчання у контрактних групах

Шляхом знаходження добутку планової кількості годин для ДКР, суми кількості студентів бюджетної форми навчання у бюджетних і контрактних групах або контрактної форми навчання у бюджетних і контрактних групах і коефіцієнта 0,33 можна обчислити загальну кількість годин, необхідних для ДКР. Коефіцієнт вказує на те, що на домашню контрольну роботу кожного студента виділяється 33% від години.

Загальна кількість годин для рефератів має бути обчислена за наступною формулою:

$$F_{заг} = F * (Б + БК) * 0{,}25,$$

де $F_{заг}$ – загальна кількість годин для рефератів,

F – планова кількість годин для рефератів,

Б – кількість студентів бюджетної форми навчання у бюджетних групах,

БК – кількість студентів бюджетної форми навчання у контрактних групах

Альтернативний варіант запису даної формули має вигляд, представлений нижче:

$$F_{заг} = F * (К + КК) * 0{,}33,$$

де $F_{заг}$ – загальна кількість годин для рефератів,

F – планова кількість годин для рефератів,

К – кількість студентів бюджетної форми навчання у бюджетних групах,

КК – кількість студентів бюджетної форми навчання у контрактних групах

З даного способу обчислення можна зробити висновок, що для обчислення загальної кількості годин для ДКР необхідно знайти добуток планової кількості годин для рефератів, суми кількості студентів на бюджетній формі навчання, що навчаються у бюджетних та контрактних групах або студентів, що навчаються на контрактній формі навчання, що навчаються у бюджетних та контрактних групах, і коефіцієнта 0,33, так як на реферат кожного студента затверджена кількість часу дорівнює 33% від години.

Наступну формулу використовують для знаходження загальної кількості годин для проведення консультацій:

$$C_{заг} = E * Г * 2 + N * \frac{(Б+БК)}{25} * k,$$

де $C_{заг}$ – загальна кількість годин для консультацій,

Г – загальна кількість груп, яким читається дисципліна,

N – загальна планова кількість годин на дисципліну,

Б – кількість студентів бюджетної форми навчання у бюджетних групах,

БК – кількість студентів бюджетної форми навчання у контрактних групах,

k – коефіцієнт відсотків залежно від форми навчання

Другий варіант виведення даної формули можна записати у наступному вигляді:

$$С_{заг} = Е * Г * 2 + N * \frac{(К+КК)}{25} * k,$$

де $С_{заг}$ – загальна кількість годин для консультацій,

Г – загальна кількість груп, яким читається дисципліна,

N – загальна планова кількість годин на дисципліну,

Б – кількість студентів бюджетної форми навчання у бюджетних групах,

БК – кількість студентів бюджетної форми навчання у контрактних групах,

k – коефіцієнт відсотків залежно від форми навчання

Дані формули демонструють, що для знаходження загальної кількості годин для проведення консультацій необхідно знайти суму двох величин, перша з яких дорівнює добутку планової кількості екзаменаційних годин, кількості груп, що вивчають дисципліну та коефіцієнту 2, друга – добутку загальної планової кількості годин на дисципліну, частки суми кількості студентів на бюджетній формі навчання у бюджетних групах або кількості студентів на контрактній формі навчання у бюджетних і контрактних групах і коефіцієнту 25, і коефіцієнту 0,06. Перший доданок формули відповідає за консультації, що проводяться перед іспитами, на які виділяється 2 години для кожної групи, у якої є іспит з даної дисципліни. Другий доданок формули відповідає за консультації з дисципліни, що проводяться з навчальних дисциплін протягом навчального семестру. З загальної кількості годин на дисципліну на кожну групу, кількість яких знаходиться діленням кількості студентів на 25, що є середнім значенням кількості людей у групі, виділяється кількість відсотків годин k, що дорівнює 2% для денної форми здобуття освіти та 6% для заочної форми здобуття освіти.

Останнім стовпець файлу заповнюється за формулою, що представлена у наступному вигляді:

$$З = Л_{заг} + П_{заг} + L_{заг} + I_{заг} + E_{заг} + Z_{заг} + M_{заг} + Q_{заг} + G_{заг} + R_{заг} + D_{заг} + F_{заг} + C_{заг},$$

де З – загальна сума годин, необхідних для усіх видів робіт з дисципліни,

$Л_{заг}$ – загальна кількість лекційних годин з дисципліни,

$П_{заг}$ – загальна кількість годин для практичних занять з дисципліни,

$L_{заг}$ – загальна кількість годин для лабораторних занять з дисципліни,

$I_{заг}$ – загальна кількість годин для індивідуальних занять з дисципліни,

$E_{заг}$ – загальна кількість годин, необхідна для проведення екзаменів,

$Z_{заг}$ – загальна кількість годин, необхідна для проведення заліків,

$M_{заг}$ – загальна кількість годин, необхідна для проведення контрольних робіт,

$Q_{заг}$ – загальна кількість годин для роботи над курсовими проєктами,

$G_{заг}$ – загальна кількість годин для роботи над курсовими роботами,

$R_{заг}$ – загальна кількість годин для РГР, РР, ГР,

$D_{заг}$ – загальна кількість годин для ДКР,

$F_{заг}$ – загальна кількість годин для рефератів.

Проаналізувавши формулу, можна зробити висновок, що загальна кількість годин з дисципліни знаходиться обчисленням суми загальних кількостей годин усіх видів робіт з цієї дисципліни.

Після того, як методист сформував файли форми К-3, починається наступний етап роботи над навчальним навантаженням кафедри. Працівник кафедри повинен розподілити усі освітні компоненти, заплановані на навчальний рік, між викладачами. Цей вид роботи працівника методичного відділу є надзвичайно важливим, адже саме під час цього виду діяльності визначається кількість годин робочих годин викладача, починає формуватися початковий варіант розкладу навчання. Цей етап вимагає максимальної уважності методиста, так як неправильно розподілене навантаження викладача може призвести до неефективного викладання дисципліни через брак часу на підготовку, некомпетентість викладача у певній навчальній дисципліні, недостачу або перебір кількості робочих годин.

Згідно з ст. 56 Закону України «Про вищу освіту» робочий час науково-педагогічних працівників становить 36 годин на тиждень і включає в себе час

виконання ним методичної, навчальної, організаційної, наукової робіт та інших трудових обов'язків.

Максимальним навчальним навантаженням науково-педагогічного працівника на одну ставку прийнято не більше 600 годин на один навчальний рік. В той час, як мінімальний обсяг у них відрізняється. Залежно від посади він становить:

- для асистентів – 550 годин;
- для викладачів, старших викладачів – 500 годин;
- для доцентів – 450 годин;
- для професорів, докторів наук – 400 годин;
- для завідувачів кафедр – 350 годин.

Якщо НПП залучений на до науково-педагогічної роботи, як сумісник на 0,5 (0,25) ставки, то робочий час таких працівник розраховується, як 18 (9) годин на тиждень, а мінімальний обсяг навчального навантаження та інших видів робіт пропорційно зменшується.

Розподіл предметів між викладачами є основною складовою частиною процесу визначення індивідуальних планів роботи науково-педагогічних працівників. Під час роботи над ними відповідальний працівник має враховувати усі особливості кожного виду роботи з метою забезпечення максимальної можливої оптимізації використання творчого потенціалу кожного НПП.

Також існують певні обмеження, які стосуються того, чи може викладач на певній посаді виконувати певний вид навчальної роботи. Лекції повинні проводитися лекторами, якими можуть бути професори і доценти. Також можливий варіант проведення лекцій провідними науковими працівниками та спеціалістами, які були запрошені для читання лекцій. З дозволу декана факультету або директора навчально-наукового інституту, де навчаються здобувачі вищої освіти, як виключення на один навчальний рік, старшому викладачеві може бути доручено проведення лекцій. Рекомендовано також доручати лекторові проведення практичних, лабораторних та семінарських занять хоча б з однією навчальною групою лекційного потоку, якому викладається дисципліна. Усі науково-педагогічні працівники, незалежно від статусу їх викладання (штатні, сумісники, з

погодинною оплатою), складають індивідуальні плани, які потім розглядає засідання кафедри та підписує її завідувач.

Навчальне навантаження науково-педагогічних працівників планується на підставі витягів з робочих навчальних планів, які випускові кафедри передають на кафедри, що викладають дані дисципліни. Саме цим витягом і є форма К-3, описана раніше у розділі.

Наукова, методична, виховна та організаційна роботи працівників плануються на навчальний рік із зазначенням конкретних підсумкових результатів у відповідних до них розділах індивідуального робочого плану.

Індивідуальний план науково-педагогічних працівник є рекомендаційним та за рішенням кафедри може змінюватись або доповнюватись.

Індивідуальні види діяльності науково-педагогічного працівника також мають певні норми. До таких видів робіт та їх норм належать:

- керівництво практиками – 2 години на один робочий день на навчальну групу з виїздом у відрядження або 6 годин без виїзду;
- проведення атестаційних екзаменів в усній формі – 0,5 години на одного здобувача вищої освіти на кожного члена комісій та її голову (не більше 6 годин на один робочий день, членів комісії не більше 4 включно з її головою);
- проведення атестаційних екзаменів у письмовій формі – до 4 годин на одну навчальну групу та 0,5 години на перевірку однієї роботи (одну роботу перевіряє один член комісії, членів комісії не більше 4 включно з її головою);
- керівництво, консультування, рецензування та проведення захисту кваліфікаційних робіт бакалаврів – 25 годин на одного здобувача вищої освіти, з яких 2 години рецензенту, до 21 години керівнику і консультантам, по 0,5 години голові комісії та кожному членові комісії (кількість членів комісії не більше 4 осіб включно з головою, плануються до 10 захистів на один день, за одним керівником закріплено не більше 8 здобувачів вищої освіти);

- керівництво, консультування, рецензування та проведення захисту кваліфікаційних робіт магістрів за ОПП – 30 годин на одного здобувача вищої освіти, з яких 3 години рецензенту, до 25 годин керівникові та консультантам, по 0,5 години кожному членові комісії та голові (кількість членів комісії не більше 4 осіб включно з головою, плануються до 8 захистів на один день, за одним керівником закріплено не більше 6 здобувачів вищої освіти);
- керівництво, консультування, рецензування та проведення захисту кваліфікаційних робіт магістрів за ОНП – 40 годин на одного здобувача вищої освіти, з яких 4 години рецензенту, до 34 годин керівнику і консультантам, по 0,5 години кожному членові комісії та голові (кількість членів комісії не більше 4 осіб включно з головою, плануються до 5 захистів на один день, за одним керівником закріплено не більше 5 здобувачів вищої освіти);
- керівництво аспірантами – 4 роки, 50 годин на кожен навчальний рік на одного аспіранта;
- наукове консультування докторантів – 2 роки, 50 годин на один навчальний рік на докторанта;
- керівництво стажуванням науково-педагогічних працівників – 8 годин на одного науково-педагогічного працівника-стажера на один місяць, не більше 30 годин на один рік;
- проведення тематичних дискусій, науково-практичних конференцій здобувачів післядипломної освіти – 1 година за одну академічну годину для кожного науково=педагогічного працівника (науково-педагогічних працівників не більше 3 осіб);
- керівництво стажуванням здобувачів післядипломної освіти – 5 годин на один тиждень на одного здобувача післядипломної освіти (проведення стажування відбувається за місцем знаходження вищого навчального закладу науково-педагогічними працівниками);
- рецензування рефератів здобувачів післядипломної освіти – 3 години на один реферат;

- керівництво, консультування, рецензування та проведення захисту випускних робіт здобувачів післядипломної освіти – до 10 годин на одну випускну роботу, з яких по 0,33 години кожному членові комісії, 2 години рецензенту (членів комісії не більше 3 осіб включно з головою);
- проведення випускних екзаменів здобувачів післядипломної освіти – 0,5 години на одного здобувача післядипломної освіти кожному членові комісії та голові (членів комісії не більше 3 осіб включно з головою).

Після завершення процесу розподілення видів усіх запланованих видів діяльностей та формування індивідуальних планів для всіх науково-педагогічних працівників методист переходить до завершального етапу процесу планування навчального навантаження кафедри. Цей етап включає в себе створення форми К-4, назва якої звучить як «План навчального навантаження науково-педагогічних працівників кафедри». Цей файл містить інформацію про:

- факультет/ННІ;
- навчальний рік, на який його складено;
- випускову кафедру;
- перелік викладачів (штатних, сумісників, з погодинною оплатою);
- перелік усіх видів діяльностей, що можуть бути закріплені за науково-педагогічними працівниками.

Файл має вигляд таблиці, де після «шапки» документу міститься список усіх викладачів. Кожен рядок відповідає за окремого викладача. У рядку містяться усі дані про види роботи, закріплені за викладачем та їх обсяг у годинах. Перелік стовпчиків кожного такого рядка:

- прізвище викладача, посада;
- частка ставки;
- кількість годин на лекції;
- кількість годин на практичні заняття (комп’ютерні практикуми, семінари);
- кількість годин на лабораторні роботи;
- екзамени;
- заліки;
- контрольні роботи (модульні, тематичні);

- курсові проєкти;
- курсові роботи;
- РГР, РР, ГР;
- ДКР;
- реферати;
- консультації;
- індивідуальні заняття (з студентами, магістрами, студентами зі змішаною формою навчання);
- керівництво практиками;
- керівництво атестаційними роботами (бакалаврів, магістрів ОПП, магістрів ОНП);
- консультування атестаційних робіт (бакалаврів, магістрів ОПП, магістрів ОНП);
- консультування з випускного екзамену (бакалаврів, магістрів ОПП, магістрів ОНП);
- рецензування атестаційних робіт (бакалаврів, магістрів ОПП, магістрів ОНП);
- робота в екзаменаційній комісії (бакалаврів, магістрів ОПП, магістрів ОНП);
- вступні екзамени (магістрів ОПП, магістрів ОНП, докторів філософії);
- керівництво (аспірантами, здобувачами післядипломної освіти, стажерами);
- консультування докторантів;
- загальна кількість годин навчального навантаження науково-педагогічного працівника;

Після завершення створення цього файлу, заповнення та перевірки усіх внесених даних роботу з навчальним навантаженням кафедри можна вважати завершеною. Надалі файли з планом навчального навантаження науково-педагогічних працівників кафедри та індивідуальні плани наукового-педагогічних працівників кафедри використовуються для планування розкладу навчального процесу на навчальний рік.

2.3 Висновки до розділу 2

Планування навчального навантаження кафедри є громіздкою та кропіткою роботою, де навіть невелика помилка у обчисленнях може призвести до абсолютно неправильних результатів у кінці. Помилки у такій великій кількості операцій з числами дуже важко відслідкувати, що часто змушує починати весь процес з самого початку. Проаналізувавши предметну область та дослідивши існуючі альтернативні програмні рішення було виявлено, що всі вони мають свої недоліки у вигляді певних програмних рішень, питань до впровадження, використання або оновлення цих систем. Також було повністю досліджено весь процес планування навчального навантаження кафедри, проаналізовано формули, які при цьому використовуються, обмеження до значень певних величин, які використовуються при плануванні.

3 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

Моделювання архітектури системи та вибір програмних засобів є найважливішим етапом розробки програмного забезпечення. Достатня кількість часу, витрачена на цей етап, допомагає правильно спланувати процес розробки, значно зменшує шанси критичних помилок під час програмування, які можуть змусити повністю переписувати код та вберігає від необхідності вставляти певні частини коду, щоб просто змусити систему працювати. У цьому розділі буде описано усі обрані засоби програмної реалізації створюваної системи, описано парадигми та методики, які будуть застосовуватись при проектуванні, та загальний опис архітектури та роботи створюваного програмного забезпечення.

3.1 Засоби програмної реалізації

Так як користувач системи, тобто працівник кафедри, може мати дуже різні умови праці та технічне обладнання, з яким він працює (старий комп'ютер, різні види ОС тощо), то було прийнято рішення зробити розроблювану системи веб-додатком, щоб користувач міг працювати з системою з будь-яких пристроїв. Мовою програмування було обрано JavaScript, так як на даний момент тільки ця технологія дозволяє створювати веб-застосунки. Надалі у підрозділі буде описано інші технології, обрані для розробки системи.

3.1.1 Редактор коду Visual Studio Code

Visual Studio Code – редактор вихідного коду, який запускається на робочому столі і доступний для Windows, macOS та Linux. Він є досить ресурсоефективним і в той самий час потужним інструментом для розробників. VS Code має вбудовану підтримку для JavaScript, TypeScript та NodeJS а також багату екосистему розширень для інших мов програмування та середовищ виконання (C++, C#, Java,

PHP тощо)[11]. Завдяки інструменту IntelliSense редактор дає можливість ефективної підсвітки синтаксису та автозаповнення, що значно полегшує та пришвидшує процес розробки. Присутня також широка можливість кастомізації: власні теми, плагіни для мов програмування, фреймворків, комбінації клавіш, що надає можливість створити індивідуальне та зручне середовище розробки[12]. Головним аргументом на користь VS Code є те, що редактор поширюється безкоштовно та є забезпеченням з відкритим кодом.

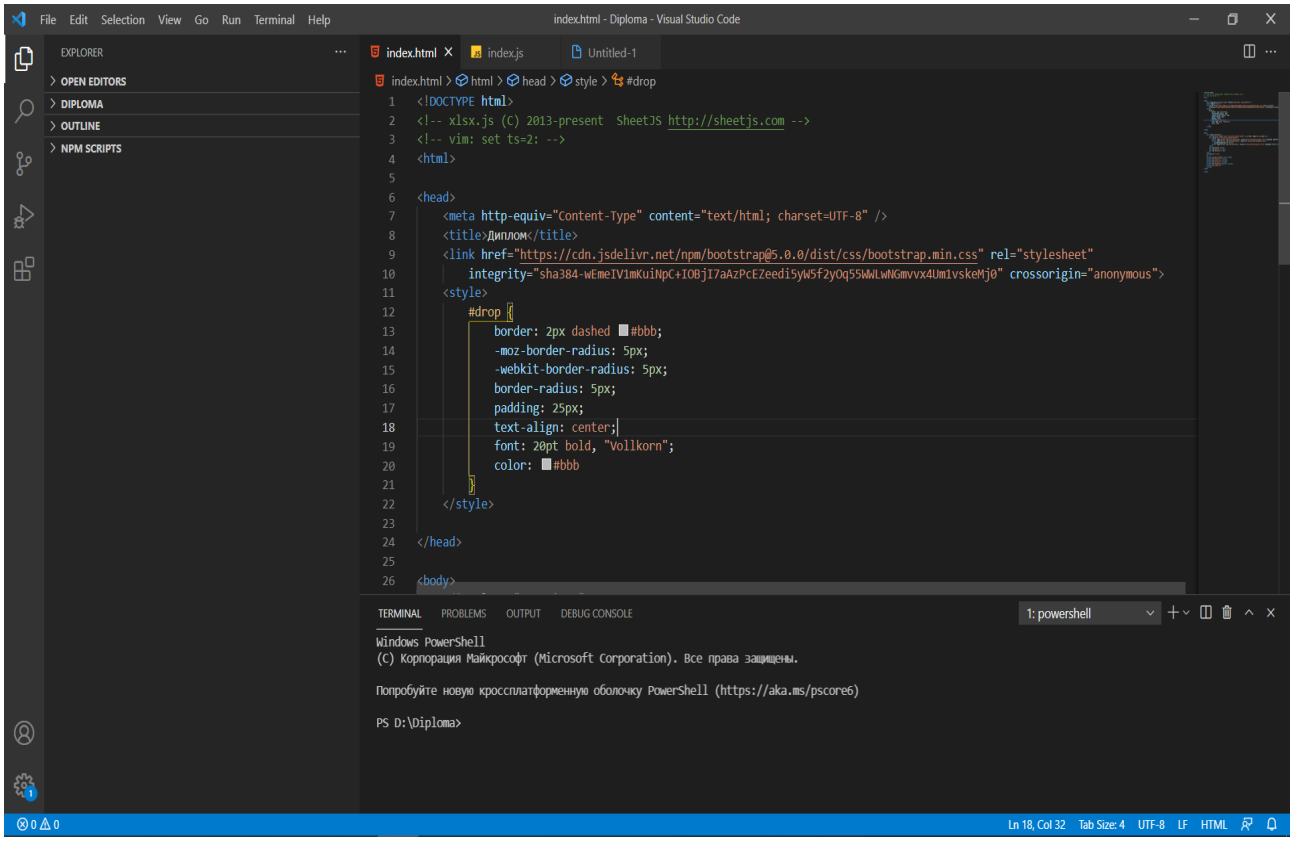

Рисунок 3.1 – Інтерфейс Visual Studio Code

3.1.2 Мова розмітки HTML

HTML (від англ. HyperText Markup Language – мова гіпертекстової розмітки) – мова розмітки, що визначає структуру веб-сторінки. HTML інтерпретується браузером; сформований відформатований текст відображається на моніторі комп'ютера або екрані мобільного пристрою. У Всесвітній павутині сторінки

HTML зазвичай передаються із сервера в браузер за допомогою HTTP або HTTPS, простого тексту або за допомогою шифрування[13].

У HTML ви можете вбудовувати код мови програмування JavaScript для контролю поведінки та вмісту веб-сторінок. Крім того, CSS, що міститься в HTML, описує зовнішній вигляд і макет сторінки. Усі HTML документи є наборами елементів, а початок і кінець елемента представлені спеціальними тегами. Елемент може бути порожнім, тобто він не містить жодного тексту чи інших даних. У цьому випадку кінцевий тег зазвичай не вказується (наприклад, тег розриву рядка
 є одиничним і не потребує закриття). Крім того елемент може мати атрибути, які визначають будь-яку з його властивостей (наприклад, атрибут href = "" у посиланні). Атрибути вказані у початковій роботі[14].

Окрім елементів, в документах HTML є англійські сутності - "спеціальні символи". Сутність починається з символу & і має форму &name; або &#NNNN;, де NNNN - код символу Unicode у десяткових записах.

Кожен документ HTML, який відповідає будь-якій версії специфікації HTML, повинен бути у версії HTML <! DOCTYPE …>

Потім початок і кінець документа позначається тегами <html> та </html> відповідно. Усередині цих тегів повинні бути теги заголовка (<head> </head>) та тіла (<body> </body>) документа.

3.1.3 Мова стилю сторінок CSS

CSS (від англ. Cascading Style Sheets - каскадні таблиці стилів) – мова таблиць стилів, що використовується для презентації документів, написаних мовою розмітки (HTML, XML). CSS, разом з HTML та JavaScript, є ключовою технологією Всесвітньої павутини[15].

Творці веб-сторінок використовують CSS для визначення кольорів, шрифтів, стилів, макету окремих блоків та інших аспектів зовнішнього вигляду цих сторінок. Основною метою розробки CSS є відокремлення опису логічної структури веб-сторінки (заповненої з використанням HTML або інших мов

розмітки) від опису зовнішнього вигляду веб-сторінки (тепер заповнена за допомогою формальної мови CSS)[16]. Цей розділ може збільшити доступність документа, забезпечити більшу гнучкість та можливість керувати його поданням, а також зменшити складність та повторюваність структурованого вмісту.

Крім того, CSS дозволяє представити один і той же документ у різних стилях або методах виводу, таких як візуалізація екрана, друк, читання голосу (спеціальний голосовий браузер чи програма зчитування з екрана) або під час виведення на пристрої, що використовують шрифт Брайля[17].

3.1.4 Мова програмування JavaScript

JavaScript (часто просто JS) – це легка, інтерпретована або JIT-компільована, об’єктно-орієнтована мова програмування з функціями першого класу. Найбільш широке застосування знаходить як мова сценаріїв веб-сторінок, але також використовується і в інших продуктах програмування, як наприклад NodeJS або Apache CouchDB[18]. JS – це прототипно-орієнтована, мультипарадигмова мова з динамічною типізацією, що підтримує об’єктно-орієнтований, імперативний та декларативний (наприклад функціональне програмування) стилі програмування[19].

Стандартом мови JavaScript є ECMAScript. Останнім релізом є ES2022, що була представлена у червні 2022, додавши новий функціонал до мови. Основними перевагами JavaScript є:

- швидкість для кінцевого користувача, так як веб-сторінки частково обробляються на стороні користувача без запитів до сервера;
- підтримуваність скриптів усіма популярними браузерами;
- потужна інфраструктура з великою кількістю фреймворків та готових рішень;
- простота у використанні — прості задачі вирішуються невеликою кількістю рядків коду, для важких завдань існує багато готових рішень, які можна адаптувати під конкретну задачу;

- зручність для створення інтерфейсів користувача, зумовлена великою кількістю обробників подій.

Недоліками мови є:

- доступність для зловмисників — у вільну скриптову мову легко додати фрагмент, що містить шкідливий код;
- нестрога типізація, що може викликати несподівані результати при приведенні різних типів даних;
- відсутність можливості знаходження помилок на ранніх етапах розробки.

В JavaScript:

- всі ідентифікатори чутливі до регістру;
- в назвах змінних можна використовувати підкреслення, літери, арабські цифри, символ долара;
- назви змінних не можуть починатися з цифри;
- для оформлення однорядкових коментарів використовуються //, багаторядкові і внутрішні коментарі починаються з / * і закінчуються * /.

Структурно JavaScript можна представити у вигляді об'єднання трьох відмінних одна від одної частин:

- ядро (ECMAScript);
- об'єктна модель браузера (Browser Object Model або BOM (англ.));
- об'єктна модель документа (Document Object Model або DOM).

Якщо розглядати JavaScript не у браузерному середовищі, то об'єктна модель браузера і об'єктна модель документа можуть не підтримуватися.

ECMAScript не є браузерною мовою, а також не визначає методи вводу та виводу. Швидше, це основа для побудови мови сценаріїв. Специфікація ECMAScript описує типи даних, інструкції, ключові слова та зарезервовані слова, оператори, об’єкти та регулярні вирази, не обмежуючи авторів похідних мов розширювати їх на нові компоненти[20].

Об'єктна модель браузера - браузер-специфічна частина мови, що є прошарком між ядром і об'єктною моделлю документа. Основне призначення об'єктної моделі браузера - управління вікнами браузера і забезпечення їх взаємодії.

Кожне з вікон браузера представляється об'єктом window, центральним об'єктом DOM.

Об'єктна модель документа - інтерфейс програмування додатків для HTML і XML-документів[21]. Згідно DOM, документ (наприклад, веб-сторінка) може бути представлений у вигляді дерева об'єктів, з багатьма властивостями, що дозволяють виконувати над ним різні операції:

- генерація і додавання вузлів;
- отримання вузлів;
- зміна вузлів;
- зміна зв'язків між вузлами;
- видалення вузлів.

3.1.5 Набір інструментів Bootstrap

Bootstrap — вільний набір інструментів для створення сайтів і веб-додатків. Включає в себе HTML- і CSS-шаблони оформлення для типографіки, веб-форм, кнопок, міток, блоків навігації та інших компонентів веб-інтерфейсу, включаючи JavaScript-розширення[22].

Bootstrap використовує сучасні напрацювання в області CSS і HTML, тому необхідно бути уважним при підтримці старих браузерів.

Перевагами використання Bootstrap для розробки веб-додатків:

- висока швидкість розробки якісної адаптивної верстки навіть розробниками-початківцями;
- кроссбраузерність та кроссплатформеність в усіх браузерах, що його підтримують;
- низький вхідний поріг — для роботи з фреймворком немає необхідності мати глибокі знання в html, css, javascript, достатньо знати хоча б основи;
- однорідність дизайну для усіх компонентів;
- велика кількість навчальних матеріалів;
- можливість налаштувань під свій проект.

Так як і більшість подібних фреймворків, Bootstrap має і недоліки:

- кінцеві файли проекту мають більший розмір, так як файли містять універсальний код, з якого потрібною для проекту є лише частина;
- складність використання bootstrap для верстки сайтів з унікальним дизайном.

Основні інструменти Bootstrap:

- сітки - заздалегідь задані розміри колонок, які можна відразу ж використовувати, наприклад, ширина колонки 140px відноситься до класу .span2 (.col-md-2 в третій версії фреймворка), який можна використовувати в CSS-описі документа;
- шаблони - фіксований або гумовий шаблон документа;
- типографіка - опис шрифтів, визначення деяких класів для шрифтів, таких як код, цитати і т. п;
- медіа - надає деяке упорядкування зображень та відео;
- таблиці - кошти оформлення таблиць, аж до додавання функціональності сортування;
- форми - класи для оформлення форм і деяких подій, що відбуваються з ними;
- навігація - класи оформлення для панелей, вкладок, переходу по сторінках, меню і панелі інструментів;
- алерт - оформлення діалогових вікон, підказок і спливаючих вікон[23].

3.1.6 Платформа Node.js

Node.js – відкрите, кросплатформове, середовище виконання для додатків, написаних мовою JavaScript, що запускаються на рушії JavaScript (наприклад рушій V8), та виконують код поза браузером[24]. Середовище було створено для побудови масштабованих мережевих додатків. Node.js дозволяє користувачеві використовувати JS для написання інструментів командного рядка та для програмування на стороні сервера – запуску скриптів на сервері, щоб створювати контент для динамічних веб-сторінок перед надсиланням їх до клієнтського браузеру. Node.js представляє парадигму «JavaScript всюди», тобто об’єднує

розробку веб-додатків навколо однієї мови програмування, а не розділяє розробку клієнтської та серверної частини з використанням різних мов.

Середовище має подійно-орієнтовану архітектуру з можливістю використання асинхронності[25]. Ці архітектурні рішення націлені на оптимізацію вводу/виводу даних та масштабованість веб-додатків з багатьма операціями вводу та виводу, а також для веб-додатків реального часу (наприклад браузерних ігор).

Node.js дозволяє створювати веб-сервери та мережеві інструменти за допомогою JavaScript і колекції «модулів», які обробляють різні основні функції. Надаються модулі для вводу/виводу файлової системи, мереж (DNS, HTTP, TCP, TLS/SSL або UDP), двійкових даних (буферів), функцій криптографії, потоків даних та інших основних функцій. Модулі Node.js використовують API, призначений для зменшення складності написання серверних програм[26].

JavaScript — це єдина мова, яку Node.js підтримує нативно, але доступно багато мов для компіляції в JS. У результаті програми Node.js можна писати на CoffeeScript, Dart, TypeScript, ClojureScript та інших. Node.js в основному використовується для створення мережевих програм, таких як веб-сервери. Найсуттєвіша відмінність між Node.js і PHP полягає в тому, що більшість функцій у PHP блокуються до завершення (команди виконуються лише після завершення попередніх команд), тоді як функції Node.js не блокуються (команди виконуються рівночасно або навіть паралельно, і використовувати зворотні виклики, щоб сигналізувати про завершення або невдачу). Node.js офіційно підтримується в Linux, macOS і Microsoft Windows 8.1 і Server 2012 (і пізніших версіях). Наданий вихідний код також може бути побудований на подібних операційних системах до тих, що офіційно підтримуються, або може бути змінений третіми сторонами для підтримки інших, таких як NonStop OS і сервери Unix[27].

V8 – рушій виконання JS, який початково був створений для Google Chrome. У 2008 році Google зробив його відритим. Написаний на C++, V8 компілює вихідний код JavaScript у власний машинний код під час виконання. Також включає у себе інтерпретатор байт-коду Ignition.

Цикл подій – це те, що дозволяє Node.js виконувати неблокуючі операції введення-виведення, незважаючи на той факт, що JS є однопоточною мовою

програмування, шляхом перевантаження операцій на ядро системи, коли це можливо. Оскільки більшість сучасних ядер є багатопоточними, вони можуть обробляти декілька операцій, що виконуються у фоновому режимі. Коли одна з операцій закінчується, ядро повідомляє Node.js, що відповідний зворотний відлік може бути доданий до черги опитування для остаточного виконання.

npm (Node Package Manager) – менеджер пакунків для мови програмування JavaScript. Для Node.js він є менеджером за замовчуванням. npm містить у собі клієнт командного рядка та онлайн базу даних публічних та платних приватних пакунків, що називається npm registry. До registry можна отримати доступ через клієнт, а доступні пакунки можна оглянути на сторінці npm. Менеджер може керувати як пакунками, що є локальними залежностями, так і глобально встановленими інструментами JS.

3.1.7 Платформа Firebase

Firebase – це платформа для розробки додатків від компанії Google, що містить у собі велику кількість сучасних інструментів для розробки[28]. Основними перевагами використання Firebase є:

- безкоштовний початковий план, що значно спрощує початок розробки додатку;
- швидкість розробки, що значно прискорюється завдяки відсутності необхідності витрачати багато часу на розробку серверної частини додатку;
- усі платформи знаходяться у одному місці, що значно полегшує інтеграцію серверних рішень у додаток. Firebase охоплює всі етапи розробки додатку та містить багато інструментів для створення, випуску та моніторингу, такі як бази даних, аутентифікація та ін.[29];
- працює на платформі Google, що зумовлює надійність системи та дозволяє інтеграцію з іншими сервісами платформи;
- дозволяє зосередитись на інтерфейсі додатку, так як надає більшість готових рішень для серверної сторони;

- надає архітектуру, у якій немає серверів, тобто оплата відбувається но основі запитів, що виключає необхідність керувати інфраструктурою серверів;
- машинне навчання поставляється комплектом API. Розробник може обирати між хмарними та вбудованими API-інтерфейсами залежно від своїх потреб;
- генерація трафіку дозволяє потенційним користувачам швидше знаходити додаток у пошуку Google[30];
- моніторинг помилок здійснюється сервісом Firebase Crashlytics, що значноє спрощує пошук та усунення помилок;
- резервне копіювання забезпечує оптимальну безпеку та доступ до даних.

Firebase використовує бази даних формату NoSQL, що є зручним у користуванні, та підходить для вирішення великої кількості задач.

3.1.8 Протокол HTTP

HTTP (HyperText Transfer Protocol) – протокол передачі даних, що використовується у комп'ютерних мережах. Головним призначенням протоколу є передача веб-сторінок та інших файлів, пов'язаних з ними (зображення та ін.)[31]. Обмін повідомленнями відбувається за схемою «запит-відповідь». Ресурси ідентифікуються за допомогою глобальних URI. Запит складається з трьох частин:

- стартовий рядок;
- заголовки;
- тіло повідомлення, у якому вказані дані запиту, ресурс або помилки, що виникли під час виконання запиту.

Структура рядку запиту виглядає наступним чином: `<Метод> <URI> HTTP/<Версія>`

Основні методи для запитів:

- OPTIONS: повертає підтримувані сервером методи HTTP, використовується для визначення можливостей сервера;
- GET: повертає вміст запитаного ресурсу;

- HEAD: робота методу аналогічна методу GET, але у відповіді сервера відсутнє тіло, використовується для отримання метаданих без пересилання вмісту;
- POST: передає дані заданому ресурсу;
- PUT: завантажує вказаний ресурс на сервер, використовується для заміни інформації на сервері;
- PATCH: завантажує частину ресурсу на сервер, використовується для зміни інформації на сервері;
- DELETE: видаляє вказаний ресурс;
- TRACE: повертає запит з можливістю для клієнта відслідкувати зміни у запиті;
- CONNECT: використовується з проксі-серверами зі здатністю перемикатись у тунельний режим SSL;

Структура першого рядку відповіді: `HTTP/<Версія> <Код статусу> <Опис статусу>`.

Класифікація кодів статусу:

- 1xx – інформаційний, вказує, що запит прийнятий;
- 2xx – успіх, вказує, що запит був успішно переданий;
- 3xx – перенаправлення, вказує, що для реалізації запиту наступні дії мають бути успішно виконані;
- 4xx – помилка клієнта, вказує, що запит не може бути виконаний або містить помилки;
- 5xx – помилка сервера, вказує, що правильний запит не виконаний сервером;

3.1.9 Бібліотека SheetJS

SheetJS – бібліотека, написана мовою програмування JavaScript, що дозволяє ефективно обробляти більшість складних електронних таблиць та генерувати нові таблиці, які будуть однаково добре працювати як зі застарілим програмним забезпечення, так і з сучасним. Бібліотека постійно оновлюється, додаються нові

формати можливих для обробки файлів. Основна робота інструментів зводиться до перетворення даних з або у масив масивів, що дозволяє максимально легко оперувати даними[32]. На даний момент бібліотека підтримує наступні формати для обробки:

- XLSB BIFF12;
- XLSX XLSM;
- SSML;
- XLS BIFF8;
- XLS BIFF5;
- XLS BIFF4;
- XLS BIFF3;
- XLS BIFF2;
- QPW;
- WQ2 WB*;
- WQ1;
- XLR;
- WKS Works;
- 123;
- WK4;
- WKS Lotus;
- UOS;
- ET;
- ETH;
- WK3;
- WK1;
- RTF;
- PRN;
- SYLK;
- DIF;
- DBF;
- TXT UTF-16;

- CSV;
- HTML Table;
- FODS;
- ODS;
- NUMBERS.

3.2 Архітектура програмного забезпечення

Вибір архітектурних рішень, як вже було визначено раніше у дослідженні, відіграє дуже важливу роль у розробці, так як дозволяє правильно націлити вектор майбутніх робіт, що надає розробникові приділяти більше часу написанню коду, а не обдумуванню, як краще буде реалізувати той чи інший функціонал. Також кожне популярне архітектурне рішення має свої загальноприйняті патерни, які рекомендується використовувати, які перевірені часом і великою кількістю досвідчених розробників. Вдале планування архітектури також значно полегшу інтеграцію зовнішніх модулів до системи і навпаки, системи, як модуля до більшого проекту. Не менш важливим етапом розробки є тестування програмного забезпечення. Саме під час покриття програми тестами та її перевірки знаходиться велика кількість неточностей та помилок у роботі програми, які здавалися неможливими під час безпосередньо розробки. У цьому підрозділі буде описано архітектуру «клієнт-сервер», принцип роботи створюваної системи та її програмну реалізацію, описано процес тестування.

3.2.1 Архітектура «клієнт-сервер»

Клієнт-серверна модель — це розподілена структура програми, яка розподіляє завдання або робочі навантаження між постачальниками ресурсів або послуг, які називаються серверами, і запитувачами послуг, які називаються клієнтами. Приклад клієнт-серверної взаємодії представлено на рисунку 3.2.

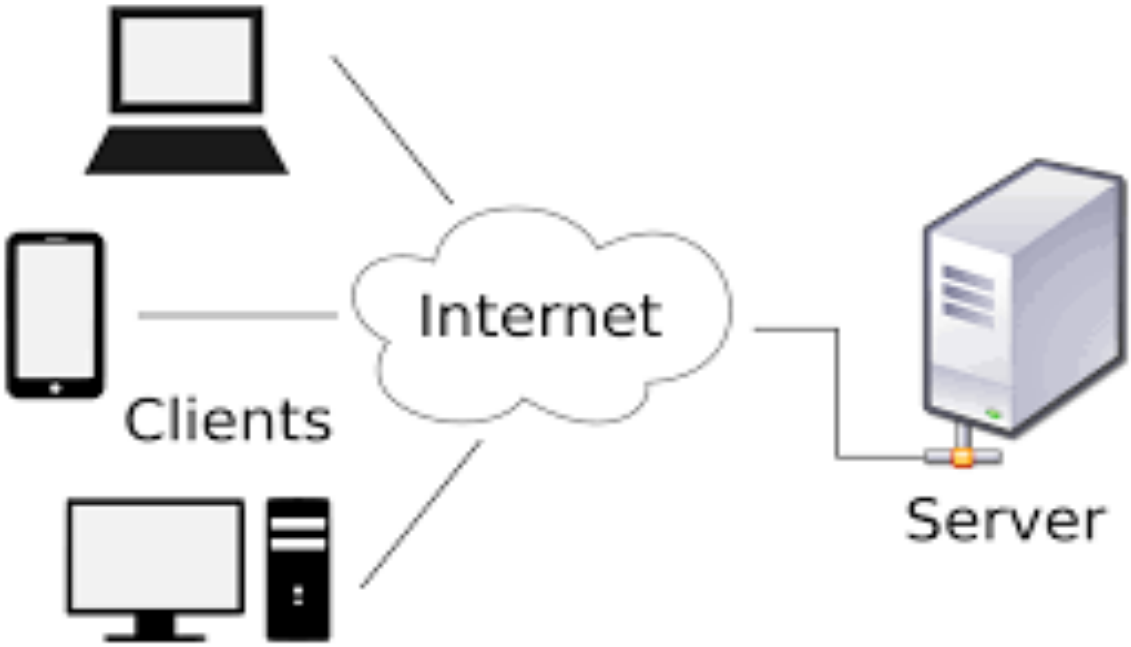

Рисунок 3.2 – Схема клієнт-серверної архітектури

Часто клієнти та сервери спілкуються через комп’ютерну мережу на окремому обладнанні, але і клієнт, і сервер можуть знаходитися в одній системі. Хост сервера запускає одну або кілька серверних програм, які спільно використовують свої ресурси з клієнтами. Клієнт зазвичай не ділиться жодними своїми ресурсами, але запитує вміст або послугу на сервері. Тому клієнти ініціюють сеанси зв’язку з серверами, які очікують вхідних запитів. Прикладами комп’ютерних програм, які використовують модель клієнт-сервер, є електронна пошта, мережевий друк і Всесвітня павутина[33].

Характеристика «клієнт-сервер» описує взаємозв'язок програм, що співпрацюють у додатку. Серверний компонент надає функцію або послугу одному або багатьом клієнтам, які ініціюють запити на такі служби. Сервери класифікуються за послугами, які вони надають. Наприклад, веб-сервер обслуговує веб-сторінки, а файловий сервер — комп’ютерні файли. Спільним ресурсом може бути будь-яке програмне забезпечення та електронні компоненти серверного комп’ютера, від програм і даних до процесорів і пристроїв зберігання. Спільне використання ресурсів сервера є послугою. Те, чи є комп’ютер клієнтом, сервером або обома, визначається характером програми, яка потребує функцій

обслуговування. Наприклад, один комп’ютер може одночасно запускати веб-сервер і програмне забезпечення файлового сервера, щоб обслуговувати різні дані клієнтам, які роблять різні типи запитів. Клієнтське програмне забезпечення також може спілкуватися з серверним програмним забезпеченням на тому самому комп’ютері. Зв’язок між серверами, наприклад для синхронізації даних, іноді називають міжсерверним або міжсерверним зв’язком. Загалом сервіс є абстракцією ресурсів комп’ютера, і клієнту не потрібно турбуватися про те, як працює сервер під час виконання запиту та доставки відповіді. Клієнт має лише зрозуміти відповідь на основі загальновідомого протоколу програми, тобто вміст і форматування даних для запитуваної послуги.

Клієнти та сервери обмінюються повідомленнями за схемою запит-відповідь. Клієнт надсилає запит, а сервер повертає відповідь. Цей обмін повідомленнями є прикладом міжпроцесної комунікації. Щоб спілкуватися, комп’ютери повинні мати спільну мову, і вони повинні дотримуватися правил, щоб і клієнт, і сервер знали, чого очікувати. Мова та правила спілкування визначаються протоколом спілкування. Усі протоколи працюють на прикладному рівні. Протокол прикладного рівня визначає основні шаблони діалогу. Щоб ще більше формалізувати обмін даними, сервер може реалізувати інтерфейс прикладного програмування (API). API — це рівень абстракції для доступу до служби. Обмежуючи спілкування певним форматом вмісту, це полегшує аналіз. Абстрагуючи доступ, він полегшує міжплатформний обмін даними.

Розроблене програмне забезпечення використовує клієнт-серверну архітектуру для розподілення інтерфейсу користувача та логіки обчислень. Під час роботи з програмою користувач завантажує файл з робочим навчальним планом, що надсилається на сервер для обробки. Також під час роботи з програмою на сервер надсилаються додаткові дані, введені на певних етапах використання системи. Вже на сервері відбувається вся робота з обробки даних, обчислення всіх необхідних параметрів для формування вихідних файлів форм К-3 та К-4.

3.2.2 Робота системи

Так як розроблена система є клієнт-серверним веб-додатком, та поки не розміщена на інтернет-ресурсі, то робота з нею починається з запуску локального серверу Node.js. Подальший сценарій роботи користувача з програмним забезпеченням представлений на діаграмі прецедентів на рисунку 3.3.

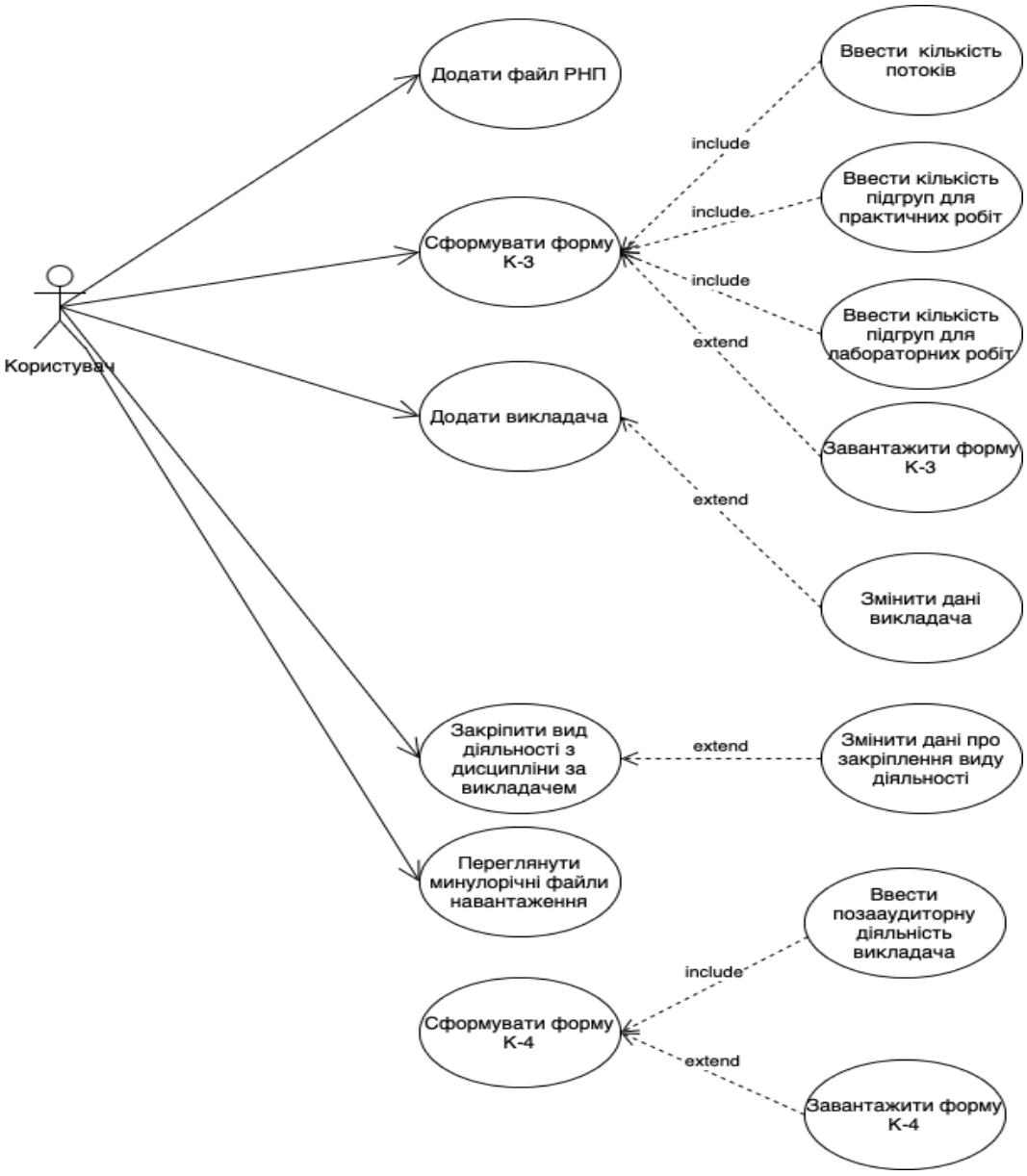

Рисунок 3.3 – Діаграма прецедентів системи

Відправною точкою роботи з системою є внесення до неї файлу з робочим навчальним планом відповідного формату. Після цього файл надсилається на

сервер для обробки. Якщо файл не проходить перевірку, користувач отримує повідомлення про те, що завантажено неправильний файл. Коли файл проходить перевірку, програма створює масив, елементами якого є масиви, що містять інформацію з кожного рядка документу. Наступним кроком є створення екземплярів класу Lesson, структуру якого зображено на рисунку 3.4.

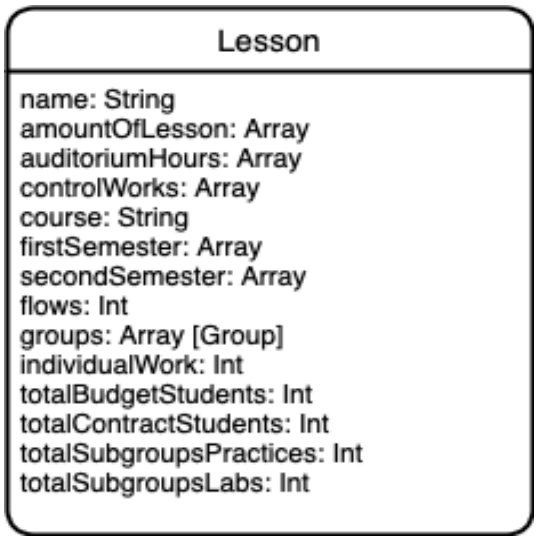

Рисунок 3.4 – Клас Lesson

Поля екземплярів заповнюються з масивів, що містять у собі рядок з назвою кафедри «ЦТЕ». Поля екземпляру класу Lesson відповідають за:

- name: назва дисципліни;
- amountOfLesson: хеш-масив, що містить пари «ключ-значення» для кількості кредитів(credits) та годин(hours);
- auditoriumHours: хеш-масив, що містить пари «ключ-значення» для загальної кількості аудиторних годин(general) та індивідуальних годин(individual), а також хеш-масиви для лекцій(lectures), практичних(practices) та лабораторних(labs) занять, кожен з яких містить інформацію про планову(plan) та індивідуальну(individual) кількість годин;
- controlWorks: хеш-масив, що містить пари для екзаменів(exams), заліків(tests), модульних контрольних робіт(modules), курсових проєктів (courseProjects) та робіт (courseWorks), РГР, РР, ГР (cgw), ДКР (hcw) та рефератів (abstracts);

- course: рядок з інформацією про курс;
- firstSemester: хеш-масив з парами про загальну кількість годин(general), лекцій(lectures), практичних(practices) та лабораторних(labs) занять;
- secondSemester: ідентичний до першого;
- flows: кількість академічних потоків;
- groups: масив екземплярів класу Group, схему якого представлено на рисунку 3.5;
- individualWork: кількість годин на індивідуальну роботу студента;
- totalBudgetStudents: загальна кількість бюджетних студентів;
- totalContractStudents: загальна кількість контрактних студентів;
- totalSubgroupsPractices: загальна кількість підгруп для практичних занять;
- totalSubgroupsLabs: загальна кількість підгруп для лабораторних занять.

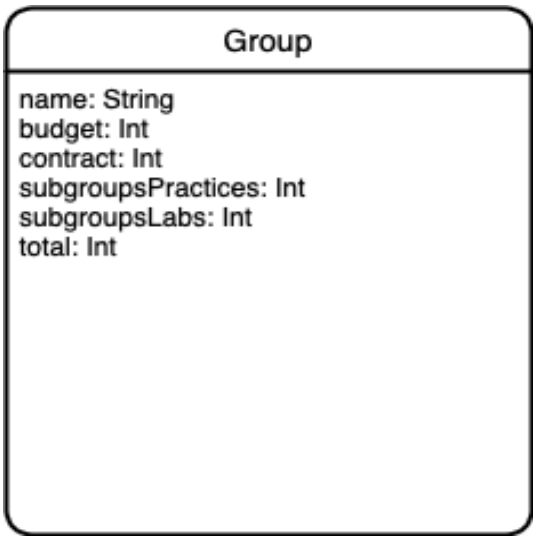

Рисунок 3.5 – Клас Group

Поля класу Group відповідають за:

- name: рядок назви групи;
- budget: кількість бюджетних студентів у групі;
- contract: кількість контрактних студентів у групі;
- subgroupsPractices: кількість підгруп для практичних занять;
- subgroupsLabs: кількість підгруп для лабораторних занять;
- total: загальна кількість студентів у групі.

Після обробки файлу системою користувачеві потрібно ввести кількість академічних потоків та кількість підгруп для груп, де кількість студентів перевищує 15 осіб. Далі система починає формувати файл форми К-3. Для цього створюється масив, елементами якого масиви з даними з кожної дисципліни. Структура масиву виглядає наступним чином:

- назва дисципліни;
- обсяг годин за семестр, N годин;
- лекції, Л годин;
- практичні заняття, П годин;
- лабораторні заняття, L годин;
- індивідуальні заняття, І;
- екзамени, Е;
- заліки, Z;
- контрольні роботи М;
- курсові проєкти, Q;
- курсові роботи, G;
- РГР, РР, ГР, R;
- ДКР, D;
- реферати, F;
- академічні бюджетні групи, Г;
- підгрупи для ПЗ., ГП;
- підгрупи для ЛР, ГL;
- академічні контрактні групи ГК;
- бюджетні студенти, Б;
- контрактні студенти, К;
- бюджетні студенти у контрактних групах, БК;
- контрактні студенти у контрактних групах, КК;
- кількість бюджетних потоків, Р;
- лекції;
- практичні заняття;
- лабораторні заняття;

- індивідуальні заняття;
- екзамени;
- заліки;
- контрольні роботи;
- курсові проєкти;
- курсові роботи;
- РГР, РР, ГР;
- ДКР;
- реферати;
- консультації;
- всього годин.

Для заповнення другої частини масиву, система проводить обчислення за формулами, описаними у розділі 2.

Після цього масив засобами бібліотеки SheetJS перетворюється у лист Excel та формується файл форми К-3. Користувач має можливість скачати даний файл, що реалізовано методами бібліотеки FileSaver.

Користувач також має можливість додати викладача до бази даних викладачів. База даних створена засобами сервісу Firebase, та є NoSQL базою даних, тобто дані зберігаються у вигляді хеш-масиву. Цей сервіс та формат БД були обрані через те, що є доволі легкими для користуваннями та маніпулюваннями даними, так як такий спосіб зберігання даних, як хеш-масив існує у більшості мов програмування. У JavaScript таким типом даних є Object. Також такий формат зберігання даних легко зрозуміє сторонній розробник, що значно полегшить покращення та підтримку системи у майбутньому, так як не при осмисленому найменуванню ключів є читабельним навіть для людини, яка не займається програмуванням. Для того, щоб додати нового викладача використовуються дані, введені користувачем, а саме ім’я викладача, його науковий ступінь, частка ставки та примітка стосовно його попереднього досвіду викладання. Метод, що використовується для цього, створює екземпляр класу Teacher, представленого на рисунку 3.6, заповнює його та відправляє дані до бази даних, де створює новий запис.

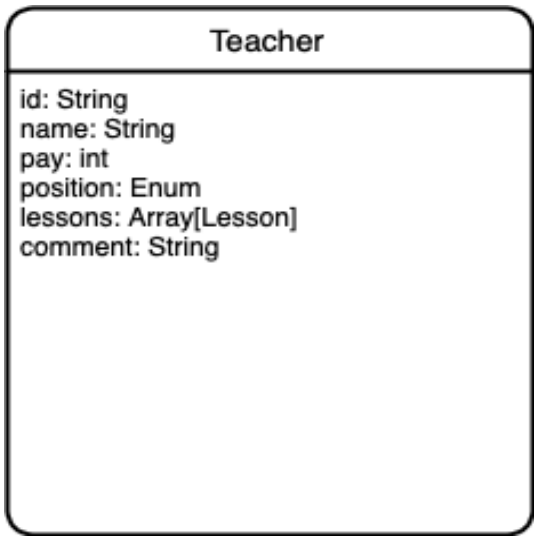

Рисунок 3.6 – Клас Teacher

Поля класу Teacher:

- id: унікальний ідентифікатор, створюється безпосередньо засобами Firebase;
- name: ім’я викладача;
- pay: частка ставки;
- position: хеш-масив з ключами: професор, доцент, старший викладач, асистент. Під кожним ключем лежить хеш-масив, що містить пари зі значеннями мінімальної кількості годин для ступеня та максимальної;
- comment: коментар стосовно попереднього викладацького досвіду, задля полегшення вибору предметів для викладача;
- lessons: масив предметів, закріплених за викладачем.

Після створення екземпляру класу, його поля конвертуються у тип даних Object, для відповідності типу зберігання даних у Firebase Realtime Database.

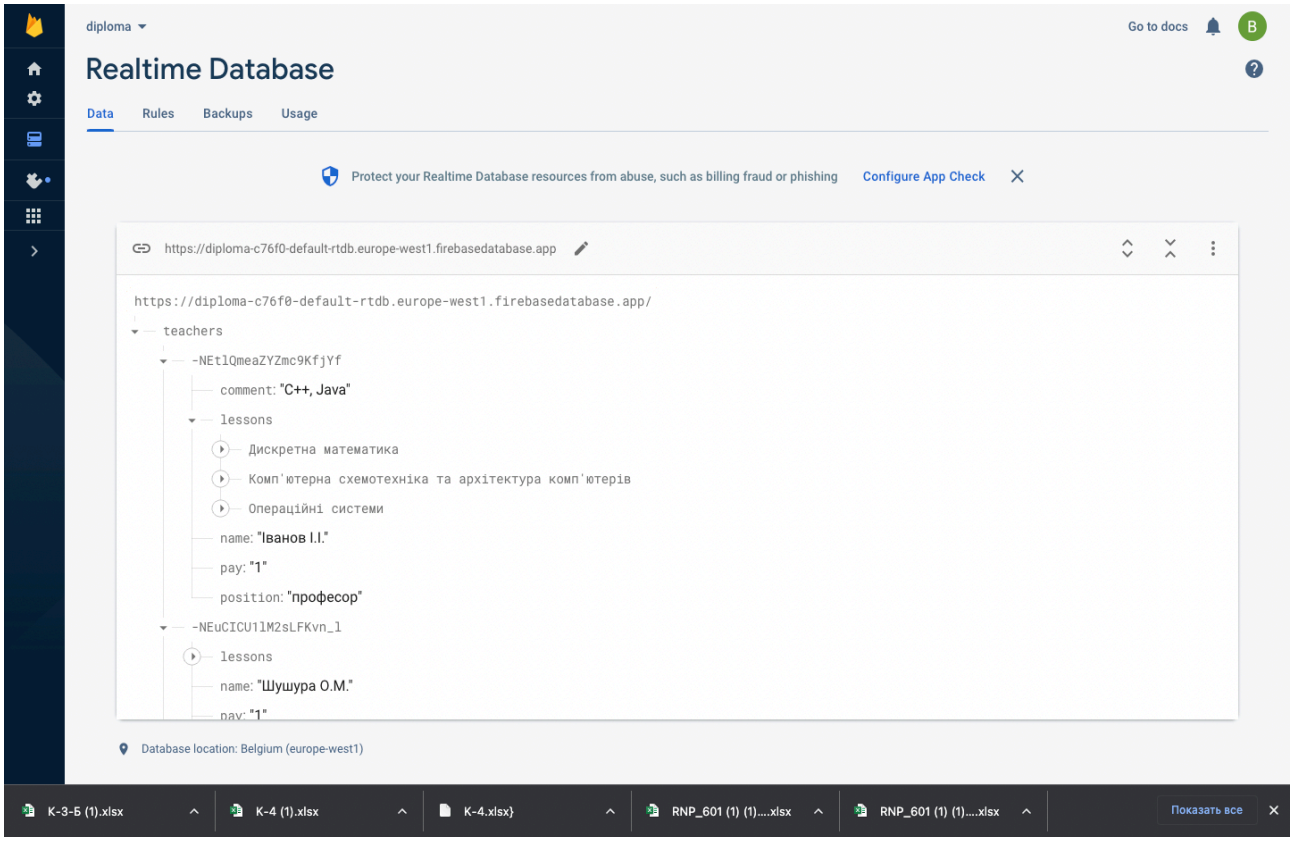

Рисунок 3.7 – База даних викладачів

На цьому етапі користувач також може змінити дані вчителя. Система надсилає запит до БД, отримує список викладачів. Після вибору викладача та внесення змін, програма виконує метод PATCH, та оновлює дані в БД за id викладача.

Наступним кроком у роботі з системою є закріплення певних видів роботи з дисципліною за викладачем. Система отримує список викладачів з бази даних, та виводить список дисциплін з усіма видами робіт. Користувач має змогу закріпити або відділити певний вид роботи за викладачем. По закінченню розподілення предметів система оновлює дані про закріплені предмети у базі даних.

Завершальним етапом роботи з системою є формування форми К-4. Користувач повинен ввести дані про позааудиторну діяльність викладача, а саме:

- індивідуальні заняття(студенти, магістри);
- керівництво практиками;
- керівництво атестаційними роботами (бакалаври, магістри ОПП і ОНП);
- консультування атестаційних робіт (бакалаври, магістри ОПП і ОНП);

- консультування по державному екзамену (бакалаври, магістри ОПП і ОНП);
- рецензування атестаційних робіт (бакалаври, магістри ОПП і ОНП);
- робота в ДЕК (бакалаври, магістри ОПП і ОНП);
- керівництво (аспіранти, стажери);
- заняття з аспірантами;
- консультування докторантів.

Після того, як вся інформація була введена та система перевірила, що жодна з величин не є пустою, дані надсилаються на сервер для формування файлу форми К-4. Користувач отримує можливість її завантажити та оновити систему для роботи з новим файлом робочого навчального плану.

Методист також має змогу обрати викладача зі списку, та сформувати індивідуальний план науково-педагогічного працівника кафедри. Після вибору викладача, дані про вибір надсилаються на сервер. За ідентифікатором відповідний викладач знаходиться у базі даних, звідки програма витягує всю необхідну інформацію про нього для формування індивідуального плану. Після заповнення масиву для створення файлу, сформований файл надсилається на клієнтський бік і користувач має змогу його завантажити.

3.3 Тестування

Unit-тести – один з обов’язкових інструментів у арсеналі будь-якого розробника ПЗ, який має на меті зробити код більш надійним та простим у обслуговуванні.

Модульне тестування – метод тестування програмного забезпечення, за використання якого створюються модулі, тобто невеликі частини застосунку, поведінка кожного з яких перевіряється окремо. Модульне тестування виконується на етапі розробки застосунку. Модулем може бути будь-який елемент коду, як, наприклад, процедура або функція[34].

Модульне тестування складається з 3 етапів:

- ініціалізація невеликої частини застосунку, яка має бути протестована;

- виклик методу, який виступає стимулом для системи, що тестується;
- спостереження за поведінкою модуля, що перевіряється. Якщо поведінка задовільняє очікування розробника, то модульний тест вважається пройденим.

Модульні тести бувають 2 видів:

- на основі стану;
- на основі взаємодії.

Якщо відслідковується отриманий стан – це модульне тестування на основі стану. Якщо тестування проходить з використанням певних методів – це модульне тестування на основі взаємодії.

Модульні тести застосовуються для перевірки різних аспектів застосунку, не витрачаючи велику кількість часу та зусиль з боку розробників. Такі тести:

- працюють, навіть якщо в системі, що тестується, є помилка. На правильні модульні тести не будуть впливати зовнішні фактори, такі як порядок виконання тощо. Якщо це відбулося, то це скоріше за все недолік дизайну;
- правильний модульний тест зрозумілий, з його допомогою легко визначити, який саме сценарій протестовано. Якщо тест не проходить, то виправити помилку можна швидко;
- модульні тести написані таким чином, що можуть використовуватися багато разів.

Модульне тестування, за правильного його використання під час розробки, значно зберігає час та гроші. До основних його переваг можна віднести:

- можливість виправлення дефектів на ранніх стадіях розробки;
- пришвидшує процес розуміння розробником кодової бази та дозволяє вчасно вносити необхідні зміни;
- можливість використання правильно написаних модульні тести замість документації;
- можливість повторного використання тестів та швидкого перенесення навіть у інший проект, за необхідності.

Як і будь-яка технологія чи методологія, unit-тести мають і недоліки. До них можна віднести:

- модульне тестування не знаходить усіх помилок у коді;
- неможливо оцінити абсолютно всі шляхи виконання, навіть у доволі тривіальних програмах;
- неможливо виявити помилки інтеграції на більш широкому рівні.

Загальним правилом модульного тестування є те, що його краще виконувати в комбінації з іншими видами тестувань для отримання більш точних результатів. Інші правила використання звучать наступним чином:

- кожен модуль повинен бути незалежним, будь-які зміни у ньому не повинні впливати на інші юніти;
- використовуються для одночасної перевірки тільки одного коду;
- потрібно завжди використовувати чіткі домовленості в іменах;
- для кожного модуля повинен бути окремий тестовий приклад;
- помилки потрібно виправляти відразу;
- чим більше коду вже написано, тим більше шляхів для перевірки на наявність помилок.

Unit-тести можуть бути як простими, так і складними в залежності від характеру об’єкта, що тестується, інструментів та стратегій, що використовуються для тестування. Але в той самий час модульні тести точно зроблять застосунок більш надійним, стабільним та функціональним.

У розробленій системі модульними тестами покриті процеси програмного обчислення величин, так як саме ці ділянки коду є критично важливими для правильності роботи програми та є найбільш вірогідним джерелом виникнення помилок.

3.4 Висновки до розділу 3

Для успішної розробки програмного забезпечення необхідно виконати дві обов’язкові умови:

- правильно змоделювати майбутню систему, підібрати підходящу архітектуру;
- обрати програмні засоби реалізації, які найкраще підходять для створення запланованого програмного продукту.

Підібрана клієнт-серверна архітектура, обрані засоби програмної реалізації та правильне тестування дали змогу розробити програмне забезпечення, що виконало усі сформовані до нього задачі.

4 РОБОТА КОРИСТУВАЧА З СИСТЕМОЮ

Для коректної взаємодії користувача з системою, він має детально розуміти можливості системи та алгоритм дій для отримання бажаних результатів. Також важливим є фактор, що на даному етапі реалізації користувач запускає застосунок на власному комп'ютері, що зобов'язує до відповідності певним вимогам. Надалі у розділі будуть описані системні вимоги, інструкції з запуску застосунку та користування ним.

4.1 Системні вимоги

Для використання розробленої системи на машині користувача повинна бути встановлена одна з наступних операційних систем:

- Microsoft Windows (XP, 7, 8, 10, 11);
- MacOS;
- Linux.

Користувач також має мати встановлений браузер з підтримкою JavaScript. Так як наразі система не розміщена на ресурсах кафедри або інших інтернет-ресурсах, то користувач вимушений розгортати сервер на власному комп'ютері. Для розгортання локального сервера користувачеві також необхідно встановити середовище Node.js, актуальну версію якого для своєї ОС можна завантажити з офіційного сайту.

Після успішного встановлення середовища, користувач у командному рядку переходить у папку з проектом, та виконує команду `node app.js`, після чого у браузері за замовчуванням відкривається сторінка стартова сторінка застосунку. Після цього користувач має змогу переходити безпосередньо до роботи з самою системою.

4.2 Сценарій роботи користувача з системою

Після запуску програми користувач бачить сторінку, зображену на рисунку 4.1. На екрані зображено 6 кнопок, назва кожної відповідає за певний етап роботи з системою. Розташовані вони у порядку їх використання.

Рисунок 4.1 – Початковий екран системи

Для початку роботи з системою, користувач натискає першу кнопку «Робота з РНП, формування К-3». Після цього під кнопкою з’являється відділ з кнопкою

«Оберіть файл», натиснувши на яку, користувач має змогу обрати файл для обробки (рисунок 4.2).

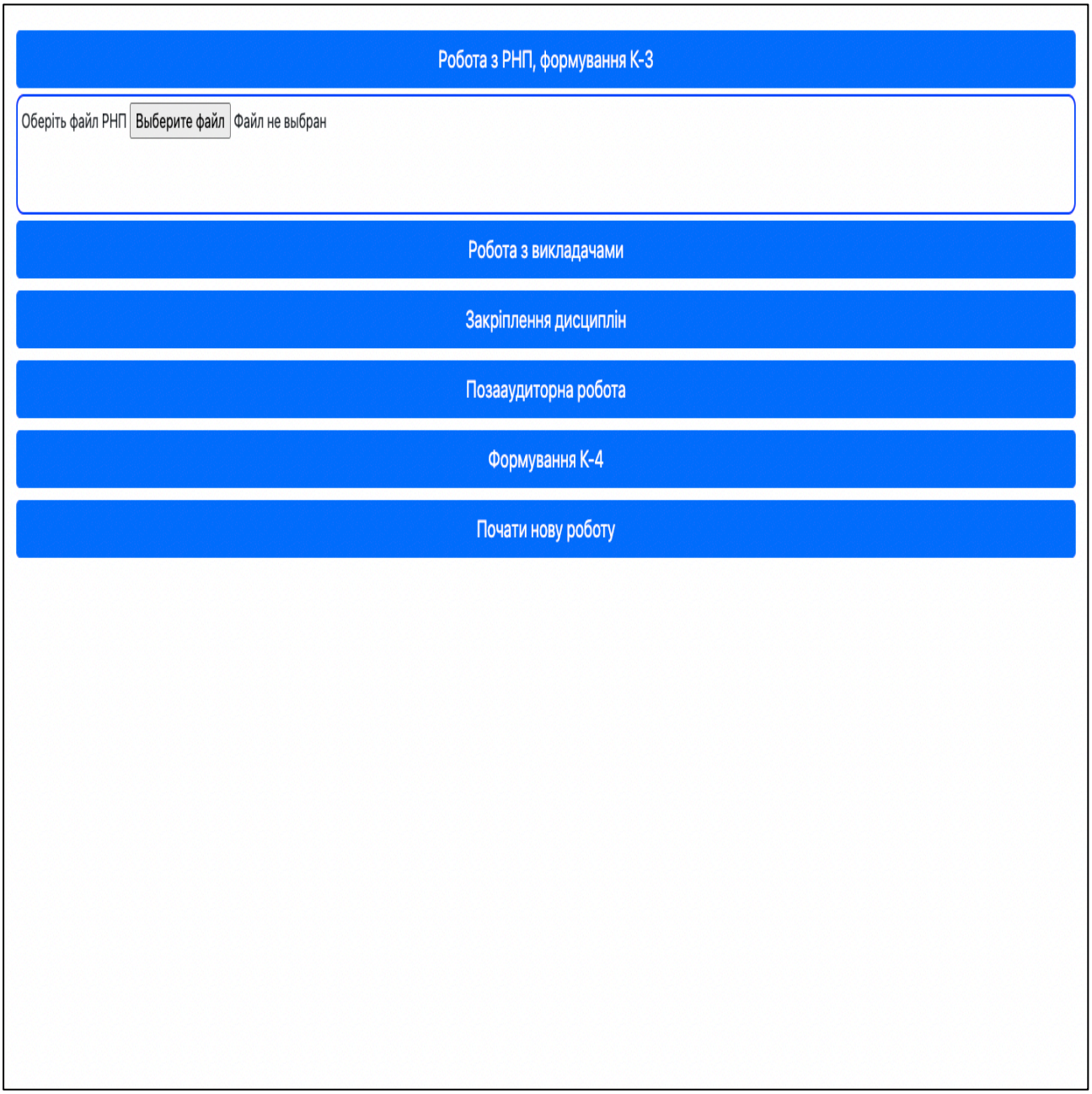

Рисунок 4.2 – Користувач обирає файл

Якщо завантажений файл не пройшов перевірку, тобто він не є файлом робочого навчального плану, то користувач отримує повідомлення, що завантажено неправильний файл (рисунок 4.3). Система також надає можливість користувачеві повторно завантажити інший файл з РНП.

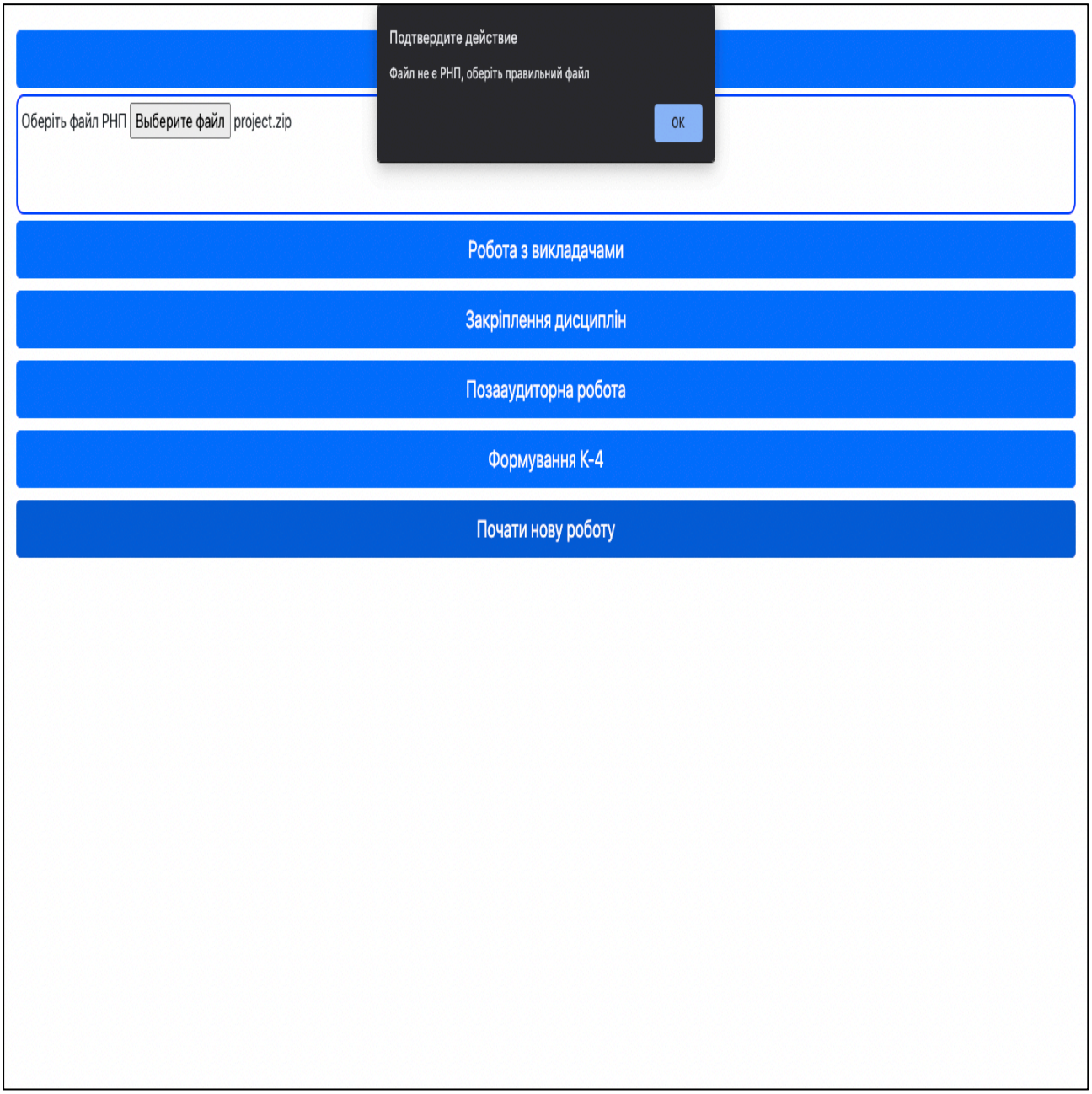

Рисунок 4.3 – Обрано неправильний файл

Коли користувач вносить правильний файл, то біля поля для вибору з’являється кнопка «Аналіз файлу» (рисунок 4.4). Це означає, що користувач завантажив у систему правильний файл, який пройшов перевірку і готовий до початку його аналізу та витягу інформації з нього.

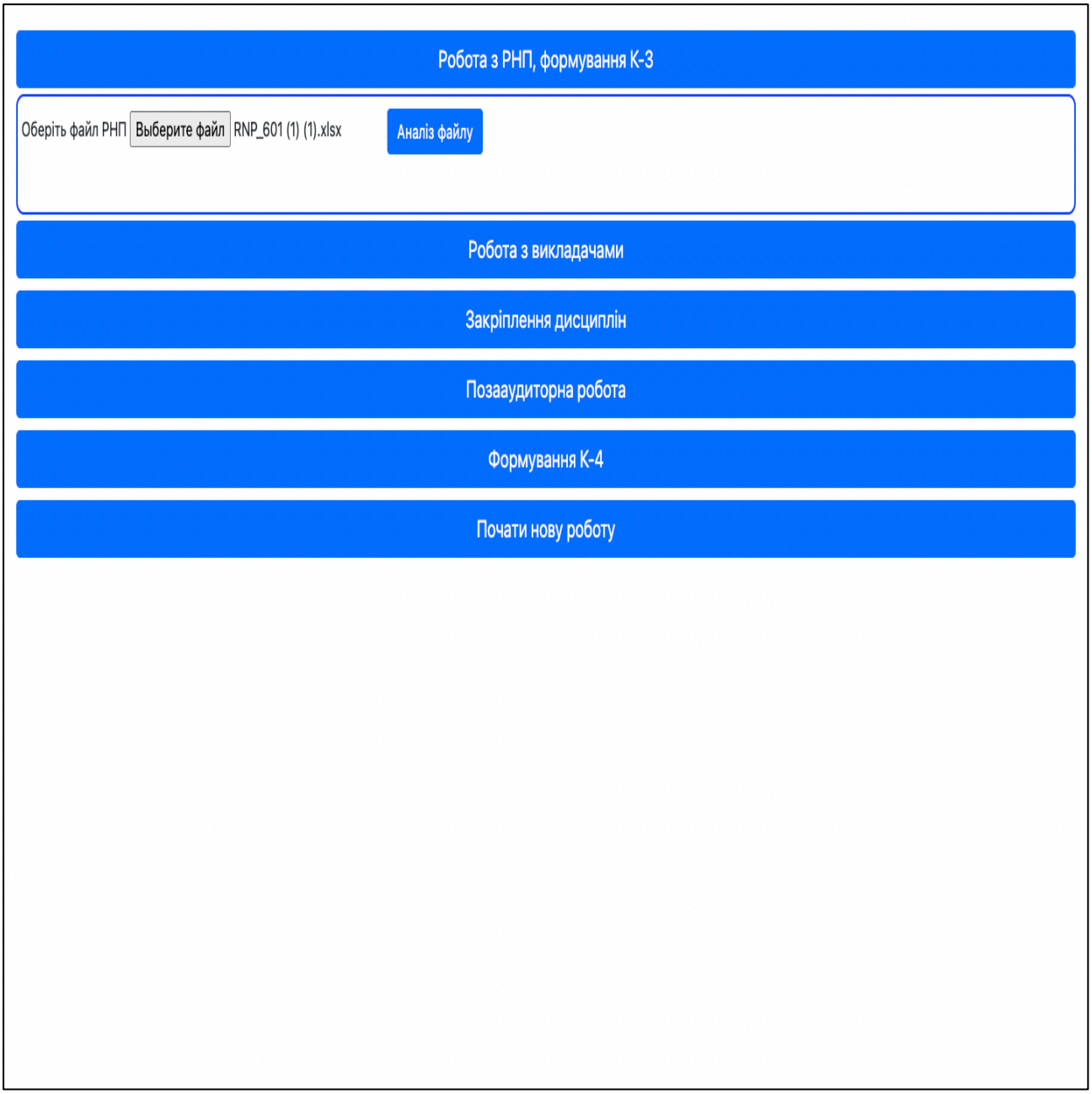

Рисунок 4.4– Файл готовий до аналізу

Після успішного аналізу системою завантаженого файлу робочого навчального плану, користувач бачить поля для введення, що з’являються на екрані (рисунок 4.5). Кількість полів може відрізнятися, так як вона залежить від кількості груп, у яких кількість студентів перевищує задані обмеження, і користувачеві необхідно вказати кількості підгруп у кожній групі.

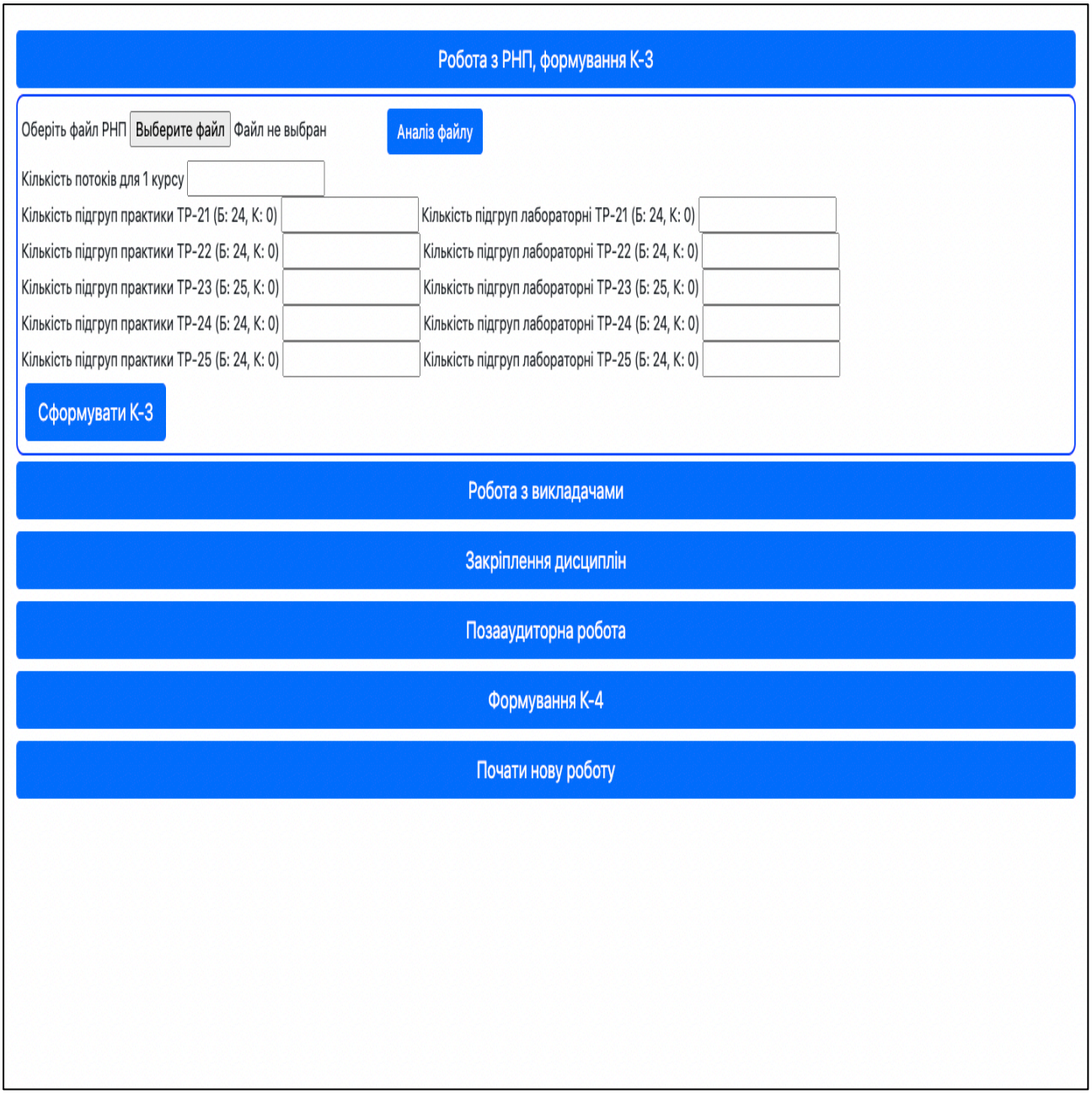

Рисунок 4.5 – Користувач вводить дані про кількість потоків та підгруп

Після правильного введення даних система користувач натискає кнопку «Сформувати К-3», після чого система завантажує на його комп’ютер відповідний файл.

Натискаючи кнопку «Робота з викладачами» користувачеві відкривається відділ з селектором викладачів та кнопками «Додати викладача» та «Змінити викладача» (рисунок 4.6).

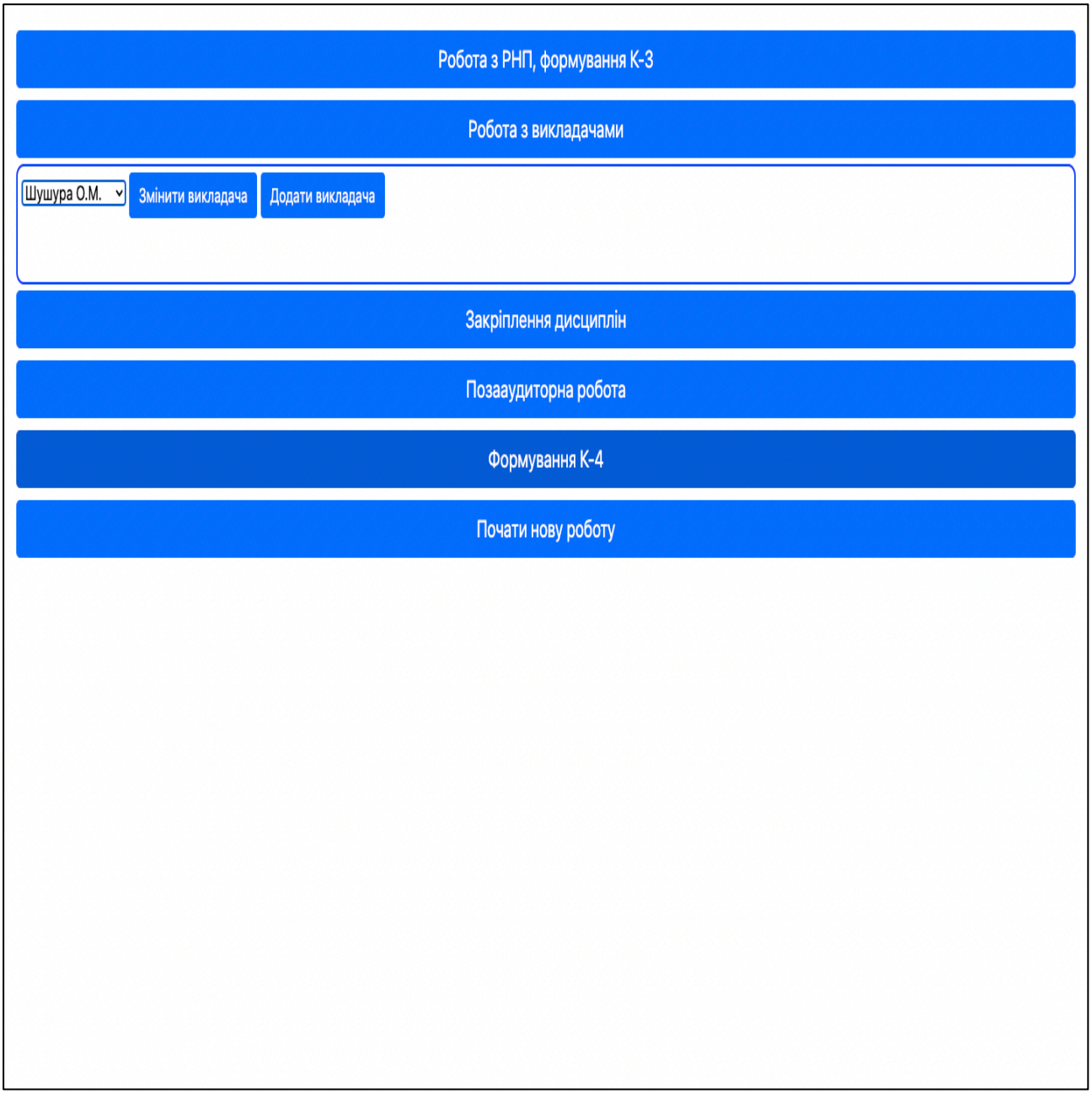

Рисунок 4.6 – Користувач працює з викладачами

При натисканні «Додати викладача» з’являються поля для введення даних нового викладача та кнопка «Додати» (рисунок 4.7). Користувач вводить дані про науково-педагогічного працівника, якого планує додати. До них входять ім’я (прізвище та ініціали), частка ставки, коментар стосовно предметів, на яких спеціалізується даний працівник та селектор з науковим ступенем. Після натискання кнопки «Додати» інформація надсилається на сервер, звідки додає викладача до бази даних.

Рисунок 4.7 – Користувач додає викладача

При натисканні «Змінити викладача» з’являються поля для зміни даних вже існуючого викладача та кнопка «Змінити» (рисунок 4.8). Користувач змінює дані про науково-педагогічного працівника, якого планує змінити. До них входять ім’я (прізвище та ініціали), частка ставки, коментар стосовно предметів, на яких спеціалізується даний працівник та селектор з науковим ступенем. Після натискання кнопки «Змінити» інформація надсилається на сервер, звідки за ідентифікатором знаходить викладача у базі даних та змінює інформацію.

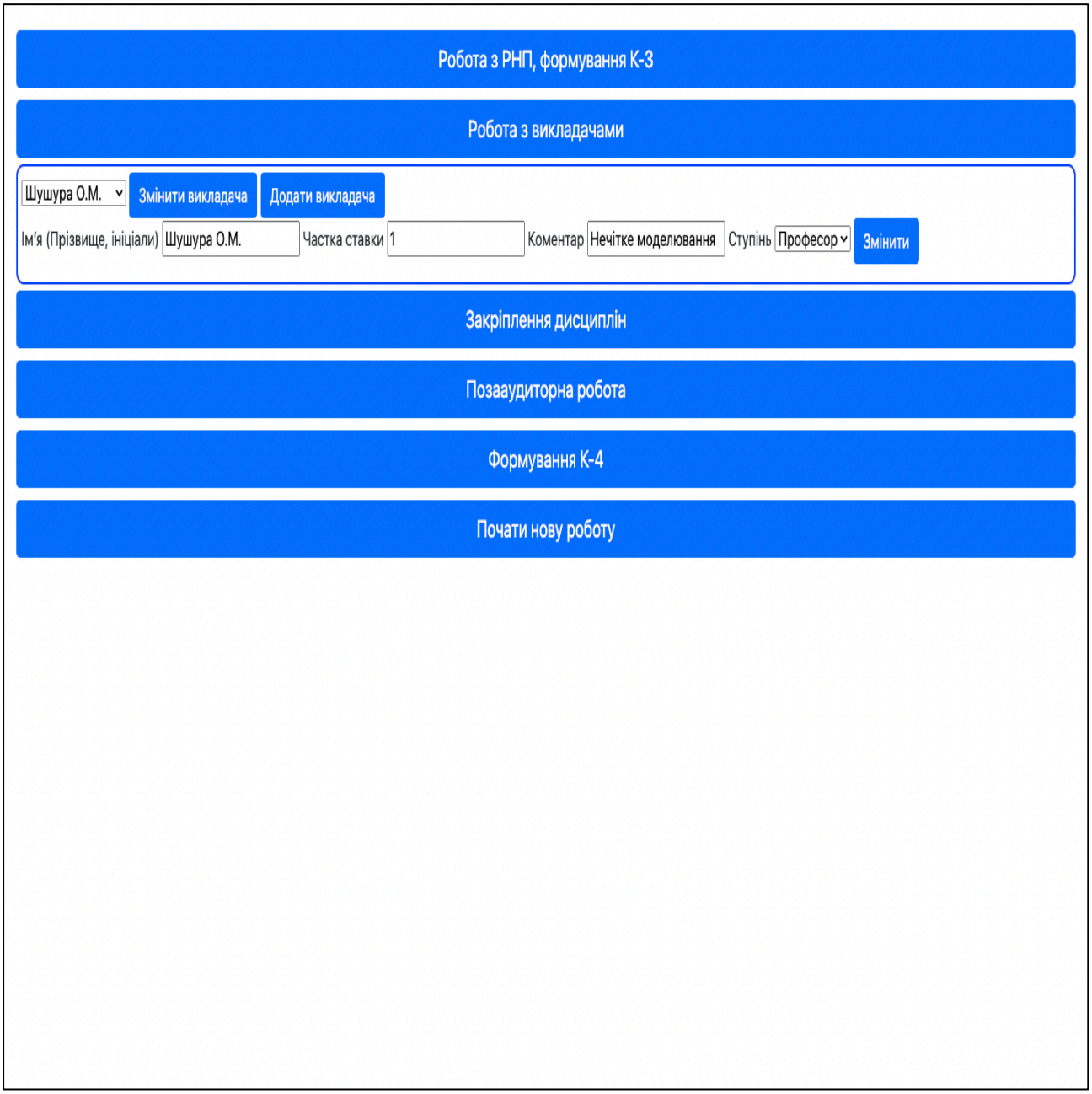

Рисунок 4.8 – Користувач змінює дані викладача

Після закінчення роботи з викладачами користувач може починати роботу над закріпленням дисциплін за викладачами. Після натискання на кнопку «Закріплення дисциплін» на екрані з’являється селектор вибору викладачів та список дисциплін з їхніми видами діяльності (рисунок 4.9). Вибравши викладача з випадаючого меню, користувач має змогу додати або видалити окремий вид діяльності з дисципліни до викладача.

Рисунок 4.9– Користувач закріпляє дисципліни

Після того, як користувач обирає викладача та натискає кнопку «Додати», біля відповідного виду діяльності з’являється прізвище та ініціали викладача, за яким він тепер закріплений (рисунок 4.10). Після натискання кнопки «Видалити», біля вілповідного виду діяльності зникає прізвище та він відкріпляється від викладача. Якщо кнопка натиснута біля ще не розподіленого виду діяльності, то нічого не відбудеться.

Рисунок 4.10– Вид діяльності закріплений за викладачем

Після натискання кнопки «Завершити» користувач може переходити до заповнення позааудиторної роботи натисканням кнопки «Позааудиторна робота». На екрані з’являються поля для введення даних про різні види робіт, селектор викладачів та кнопки «Додати» та «Завершити» (рисунок 4.11). Користувач заповнює усі поля для введення (якщо діяльність незапланована, вводиться 0), після чого натискає кнопку «Додати», що оновлює інформацію у базі даних стосовно викладача.

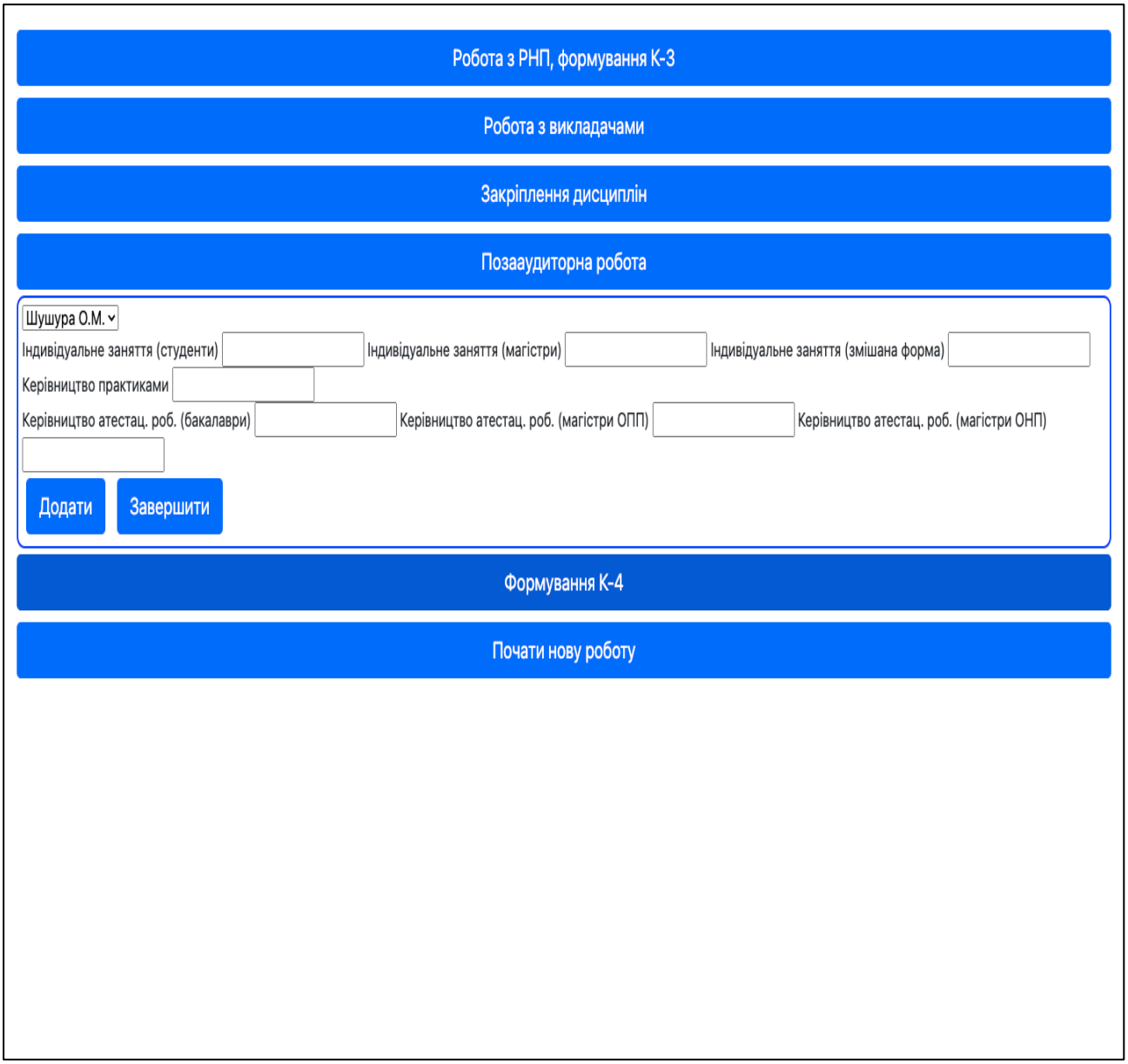

Рисунок 4.11 – Користувач додає позааудиторну роботу

Після натискання кнопки «Завершити» користувач має змогу почати нову роботу, натиснувши «Почати нову роботу», щоб опрацювати новий РНП, або завантажити актуальну версію файлу форми К-4, натиснувши «Формування К-4». Остання завантажить на комп’ютер користувача файл.

На цьому робота користувача з системою завершується. Він може закрити командний рядок, щоб зупинити роботу серверу.

4.3 Висновки до розділу 4

У розділі наведено системні вимоги до створеного програмного забезпечення та описано повний процес взаємодії користувача з системою. Під час роботи над розділом було виявлено, що додаток працює стабільно, інтерфейс є досить зрозумілим та зручним у користуванні.

Користувач надано можливість виконати всі задачі, що стоять перед ним у процесі планування навчального навантаження кафедри. Він може витягувати необхідні дані з робочого навчального плану, формувати файли форми К-3, планувати індивідуальне навантаження науково-педагогічного працівника та кафедри загалом.

5 РОЗРОБКА СТАРТАП-ПРОЄКТУ

Розділ має на меті проведення маркетингового аналізу стартап проекту задля визначення принципової можливості його ринкового впровадження та можливих напрямів реалізації цього впровадження [35].

5.1 Опис ідеї системи

Розвиток технологій, що переживає сучасний світ, значно полегшує життя людства у всіх сферах. Та реальність показує, що за покращенням слідує неперервне підвищення вимог, яке пов’язане з інтеграцією технологій у різні сфери. Різні компанії змушені впроваджувати інноваційні підходи автоматизації праці, щоб мати змогу залишатися конкурентоспроможними і не витрачати час працівників на рутинну роботу. Ця проблема дійшла і до навчальних закладів, де велика кількість роботи, пов’язаної з документообігом, досі виконується виключно працівниками, з мінімальною кількістю автоматизації процесів, яка найчастіше створена силами самих працівників. Однією з таких громіздких задач є формування навчального навантаження кафедри. Цей процес вимагає від відповідального за нього виключної уваги та зосередженості. Так як цей процес є доволі рутинним і пов’язаний в більшості з обчисленням певних величин, це значно підвищує шанси помилки у обчисленнях через одноманітність роботи та важкість знаходження помилки через велику кількість розрахунків. Задля автоматизації цього процесу було проведено дане дослідження та розроблено програмний продукт.

Ретельно проаналізувавши потреби методистів вищих навчальних закладів, можна скласти наступну таблицю, яка характеризує зміст ідеї, напрямки застосування, а також вигоди для кожного типу користувача (таблиця 5.1).

Таблиця 5.1

Опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Створення системи для автоматизації планування навчального навантаження кафедри	1. Система як джерело інформації про навчальне навантаження науково-педагогічних працівників. 1. Система як засіб формування необхідної документації	1. Система повинна бути зрозумілою та зручною у користуванні. 2. Система повинна надавати користувачеві можливість перевіряти інформацію про поточний стан навантаження викладачів. 3. Система повинна надавати можливість методистам формувати необхідні документи.

Наступним кроком було проведення аналізу наявних рішень в галузі, їх огляд та порівняння з запропонованим рішенням (таблиця 5.2).

Ринок українського програмного забезпечення пропонує наступні рішення управління навчальним процесом:

Таблиця 5.2

Визначення сильних, слабких та нейтральних характеристик ідеї проекту

№ п/п	Техніко-економічні характеристики ідеї	(потенційні) товари/концепції конкурентів			W (слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
		Мій проект	АСУ «ВНЗ»	«Політек-СОФТ»			
1.	Технологічність: 1. База даних. 2. Мова програмування. 3. Використання бібліотеки.	1. Firebase Realtime Database. 2. JavaScript. 3. SheetJs.	1. SQL. 2. Delphi. 3. ADO, Midas.	1. Interbase 6.0, Firebird. 2. Delphi. 3. Data Access, Additional.	1. – 2. – 3. –	1. – 2. – 3. –	1. Використання хмарного рішення, яке не вимагає зовнішнього адміністрування, доступність, надійність, швидка інтеграція у проект. 2. Сучасна мова

Продовження таблиці 5.2

							програмування, висока швидкість розробки. 3. Сучасна потужна бібліотека для парсингу файлів.
2.	Автономність	Не потребує зовнішньої підтримки з боку розробника.	Потребує окремого розробника.	Для налаштування мережі здійснюється звернення до послуг системного адміністратора.	-	-	Система може підтримуватися розробниками без залучення зовнішніх спеціалістів.
3.	Кросплатформеність	Веб-інтерфейс для будь-	Складається з підсисте	Складається з підсисте	-	Система являє собою	Зручний спосіб користуван

Продовження таблиці 5.2

		якого веб-браузера.	м.	м, з можливістю генерації веб-сторінок.		рішення саме для задачі планування навчального навантаження кафедри.	ня застосунком, що не вимагає потужного апаратного забезпечення.
4.	Гнучкість	Програма досить легко редагується під вимоги іншої кафедри.	Оновлення та підтримки після купівлі системи не передбачено.	Можливо підключити додаткові пакети.	-	Необхідність у наявності кваліфікованих розробників для підтримки гнучкості.	Відсутність необхідності залучення додаткових спеціалістів, наявна можливість модифікації системи без повного перепланування.

Кінець таблиці 5.2

5.	Економічність	При розміщенні на серверах навчального закладу або кафедри не потребує купівлі хостингу.	Впроваджувалася в закладах вищої освіти безкоштовно до 2006 року (за умовами партнерства).	Програмні модулі оплачуються окремо відповідно до обраного функціоналу.	Теоретично може коштувати дорожче, ніж ная вні ана лог и.	Не передбачається додаткових витрат окрім виплат розробникам.	Зручність функціоналу ймовірно виправдає витрати на розробку.

Після проведення аналізу слабких, нейтральних і сильних сторін ідеї потенційного товару було зроблено висновок, що дана система може вважатися конкурентоспроможною в умовах поставленого завдання.

5.2 Технологічний аудит ідеї проєкту

Щоб визначити, чи ідея проєкту є технологічно здійсненною, було проведено аналіз наступний складових (таблиця 5.3):

- технології, за якими буде реалізовано товар згідно ідеї проєкту;
- існування таких технологій, необхідність їх розробки/доопрацювання;
- доступність таких технологій авторам проєкту.

Таблиця 5.3

Технологічна здійсненність ідеї проєкту

№ п/п	Ідея проєкту	Технології її реалізації	Наявність технологій	Доступність технологій
1.	Парсинг робочого навчального плану	JavaScript, SheetJS	Наявна технологія	Доступна
2.	Створення веб-інтерфейсу	HTML/CSS, JavaScript	Наявна технологія	Доступна
3.	Формування форми К-3	JavaScript, SheetJS	Наявна технологія	Доступна
4.	Формування плану розподілу навчального навантаження кафедри		Наявна тезнологія	Доступна
5.	Формування індивідуального навантаження науково-педагогічного працівника	JavaScript, SheetJS, Firebase	Наявна технологія	Доступна

З даних, поданих у таблиці можна зробити висновок, що з технологічної точки зору реалізація системи з запланованим функціоналом повністю здійсненна.

5.3 Аналіз ринкових можливостей запуску стартап-проєкту

Аналіз потенційного ринку стартап-проєкту та його опис є наступним кроком. До цього процесу входять наявність попиту, обсяг і динаміка розвитку ринку (таблиця 5.4).

Таблиця 5.4

Попередня характеристика потенційного ринку стартап-проєкту

№ п/п	Показники стану ринку (найменування)	Характеристика
1.	Кількість головних гравців, од	1 головний гравець
2.	Загальний обсяг продажів, грн/ум.од	$1 млн
3.	Динаміка ринку	низька
4.	Наявність обмежень для входу (вказати характер обмежень)	Затвердження ліцензійних умов для впровадження, формування маркетингової стратегії для ефективного проведення рекламної кампанії
5.	Специфічні вимоги до стандартизації та сертифікації	Відстуні
6.	Середня норма рентабельності в галузі (або по ринку), %	+15%

За результатами аналізу потенційного ринку було зроблено висновок, що через рентабельність ринку (+15%) інвестування у даний проєкт є вигідним. За попереднім оцінюванням, ринок є привабливим для входу.

Наступним кроком є визначення потенційних груп клієнтів, їх характеристик та формування орієнтовного переліку вимог до товару для кожної з груп (таблиця 5.5).

Таблиця 5.5

Характеристика потенційних клієнтів стартап-проєкту

№ п/п	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
1.	Створення системи для автоматизації планування навчального навантаження кафедри	Методисти кафедри ЦТЕ НН ІАТЕ НТУУ «КПІ ім Ігоря Сікорського», методичні відділи вищих навчальних закладів	Методисти кафедри ЦТЕ потребують сучасної автоматизованої системи планування навчального навантаження кафедри, працівники інших ВНЗ також можуть бути зацікавлені у даному програмному продукті.	Система повинна забезпечити зручний застосунок для повного циклу процесу планування навчального навантаження кафедри.

Визначивши потенційні групи клієнтів, необхідно провести аналіз ринкового

середовища і скласти таблицю факторів загроз (таблиця 5.6).

Таблиця 5.6

Фактори загроз

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
1.	Вибір аналогу розробленого програмного забезпечення	Керівництво кафедри може обрати вже готовий програмний продукт іншого розробника.	Продовження розробки програмного продукту, так як його можна інтегрувати і в інших навчальних закладах.
2.	Планування та послідовність виконання задач	Неправильне планування та розподіл задач при розробці продукту декількома виконавцями.	Перевірка плану розробки та розподілу задач між розробниками для ефективної роботи кожного з виконавців.

Наступним етапом є аналіз факторів можливостей, що сприяють впровадженню проєкту (таблиця 5.7).

Таблиця 5.7

Фактори можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1.	Аналіз досвіду конкурентів	Розроблення стратегії для реалізації проєкту шляхом аналізу помилок конкурентів	Планування і реалізація проєкту, намагаючись у максимально можливій мірі не припускатись помилок, здійснених конкурентами.

Продовження таблиці 5.7

2.	Пріоритет на відсутню у конкурентів функціональність	Впровадження нових можливостей для користувачів, додавання нового функціоналу	Глибокий аналіз продуктів конкурентів, чітке планування задач та їх розподіл між виконавцями.
3.	Підвищення рентабельності	Можливість залучити до розробки студентів старших курсів	Зменшення суми для інвестицій у розробку та впровадження продукту.

Після аналізу можливостей було проведено ступеневий аналіз конкуренції на ринку (таблиця 5.8).

Таблиця 5.8

Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентноспроможною)
1. Олігополістична конкуренція	В загальному галузь є конкурентною, але присутні декілька явних лідерів	Вихід на міжнародний рівень є ускладненим
2. Локальний рівень конкурентної боротьби	Українські конкуренти	Розвиток в українській IT сфері та вихід на ринок
3. Внутрішньогалузева конкуренція	Конкуренцію можна помітити у пропозиціях на	Розробка ПЗ, що є вузьконаправленим

	купівлю ПЗ та якості	

Продовження таблиці 5.8

	пропонованого функціоналу	
4. Товарно-видова конкуренція	Конкуренція між програмними продуктами одного виду	Розробка більш якісних варіантів програмного забезпечення, аналіз відгуків клієнтів.
5. Нецінова конкуренція	Функціонал програмного продукту	Розширення спектру можливостей програмного забезпечення.
6. Немарочна конкуренція	Значущість якості розробки	Розробка надійного та ефективного програмного продукту.

Наступним кроком є детальніший аналіз умов конкуренції за моделлю М. Портера (таблиця 5.9).

Таблиця 5.9

Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
	1. АСУ «ВНЗ» 2. ПП «Політек	- Можливість обрати вже існуючі	Як основних постачальників можна зазначити літературу з	Методисти кафедри ЦТЕ НН ІАТЕ НТУУ	Наявні лише універсальні рішення, жодне з

	-СОФТ»	програмні			

Продовження таблиці 5.9

		продукти, нечітке ТЗ	Програмування та інтернет-ресурси	«КПІ ім.Ігоря Сікорського», співробітники інших ВНЗ.	яких не задовільняє повністю потреби методистів.
Висновки:	Конкурента боротьба не є інтенсивною через зосередженість конкурентів на іншій функціональності.	Можливість входу на ринок через більш ймовірне досягнення цілей, до потенційних конкурентів відносяться вже існуючі компанії.	На розробку даного продукту постачальники не мають впливу, так як розробка не потребує платних засобів розробки.	Програмний продукт створюється спеціалізовано для «КПІ», хоча може бути застосований і у інших ВНЗ, влада клієнтів є значною через специфіч	Абсолютні товари-замінники відсутні, присутні лише універсальні рішення.

				нічсть проєкту.	

З проведеного аналізу можна зробити висновок, що ймовірність успіху при вході на ринок є досить високою. Сильні сторони системи є конкурентоспроможними.

На основі аналізу конкуренції із урахуванням характеристик ідеї проєкту, вимог споживачів до продукту та факторів маркетингового середовища можна визначити перелік факторів конкурентоспроможності (таблиця 5.10).

Таблиця 5.10

Обґрунтування факторів конкурентоспроможності

№ п/п	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проєктів значущим)
1.	Потреби споживачів	Потреби споживачів обумовлюють необхідність розробки продукту. Якісно проведений аналіз дозволяє створити цілісне і точне ТЗ.
2.	Результативність	Досягнення кінцевого результату, кожен проміжний результат є цілісною та робочою версією продукту.
3.	Ціна та собівартість	Використання сучасних та безкоштовних засобів розробки дозволяє виставити нижчу ринкову ціну.
4.	Технічне обслуговування	Постійне оновлення програмного продукту.

За визначеними у попередній таблиці факторами конкурентоспроможності здійснюється аналіз сильних та слабких сторін стартап-проєкту (таблиця 5.11).

Таблиця 5.11

Порівняльний аналіз сильних та слабких сторін системи автоматизації планування навчального навантаження кафедри

№ п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні з системою автоматизації планування навчального навантаження кафедри						
			-3	-2	-1	0	1	2	3
1.	Потреби споживачів	7		+					
2.	Результативність	15				+			
3.	Ціна та собівартість продукту	10			+				
4.	Технічне обслуговування	8		+					

Фінальним етапом ринкового аналізу можливостей впровадження проєкту є складання SWOT-аналізу (матриці аналізу сильних (Strength) та слабких (Weak) сторін, загроз (Troubles) та можливостей (Opportunities) (таблиця 5.12), на основі виділених ринкових загроз та можливостей, сильних і слабких сторін.

Таблиця 5.12

SWOT-аналіз стартап-проєкту

Сильні сторони: Кросплатформеність, зручний та зрозумілий у використанні інтерфейс користувача..	Слабкі сторони: Можливі баги у початкових версіях продукту.
Можливості: Розширення функціоналу, вихід на національний ринок.	Загрози: Недопрацювання на початкових етапах може відштовхнкти клієнтів.

За результатами SWOT-аналізу розробляються альтернативні заходи ринкової поведінки для виведення стартапу на ринок та оптимальний час реалізації, зважаючи на продукти конкурентів.

Після визначення альтернатив проводиться аналіз строків реалізації та ймовірності отримання ресурсів (таблиця 5.13).

Таблиця 5.13

Альтернативи ринкового впровадження стартап-проєкту

№ п/п	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1.	Створення система як автономного продукту, що буде функціонувати лише в «КПІ».	85%	8 міс.
2.	Створення системи як універсальної, що буде функціонувати в будь-якому ВНЗ.	60%	12 міс.
3.	Створення системи на основі існуючих рішень з модифікаціями.	25%	5 міс.

Після аналізу альтернатив найбільш доцільною була обрана перша, так як вона є найбільш оптимальною за часом та ймовірністю отримання ресурсів.

5.4 Розроблення ринкової стратегії проєкту

Першим кроком у розробці ринкової стратегії передбачається визначення стратегії охоплення ринку, а саме опис цільових груп потенційних споживачів (таблиця 5.14).

Таблиця 5.14

Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1.	Університет «КПІ»	Готові	високий	Мала конкуренція	Легкий
2.	ВНЗ	Готові	середній	Невелика конкуренція	Легкий

Основною цільовою групою обрано НТУУ «КПІ ім. Ігоря Сікорського», так як саме для цієї групи споживачів є можливість повністю задовольнити вимоги до програмного продукту. У групи є високий попит та майже відсутня конкуренція, простота входу у сегмент – легка.

За результатами аналізу потенційних груп споживачів необхідно обрати цільові групи, яким буде запропоновано продукт, та визначається стратегія охоплення ринку. Також для роботи в обраному сегменті необхідно сформувати базову стратегію розвитку (таблиця 5.15).

Таблиця 5.15

Визначення базової стратегії розвитку

№ п/п	Обрана альтернатива розвитку проєкту	Стратегія охоплення ринку	Ключові конкурентноспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
1.	Створення системи як	Стратегія концентрованого маркетингу	Просте впровадження системи, низька ціна, максимальна	Стратегія спеціалізації

Продовження таблиці 5.15

	автономної, для функціонування лише в НТУУ «КПІ ім. Ігоря Сікорського»		відповідність потребам споживачів.	

Наступним кроком є вибір стратегії конкурентної поведінки (таблиця 5.16).

Таблиця 5.16

Визначення базової стратегії конкурентної поведінки

№ п/п	Чи є проєкт «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки?
1.	Проєкт не є «першопрохідцем», на даний момент є декілька рішень, що є універсальними для всіх ВНЗ.	Компанія буде як забирати існуючих споживачів у конкурентів, так і шукати нових.	Основні характеристики товару будуть частково схожими на характеристики конкурентів, але значно специфічнішими.	Стратегія заняття конкурентної ніші.

На основі вимог споживачів з обраних сегментів до постачальника та до продукту, а також в залежності від обраної базової стратегії розвитку та стратегії конкурентної поведінки розробляється стратегія позиціонування (таблиця 5.17), що

полягає у формуванні ринкової позиції, за якою споживачі будуть ідентифікувати проєкт.

Таблиця 5.17

Вибір стратегії позиціонування

№ п/п	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентноспроможні позиції власного стартап-проєкту	Вибір асоціацій, які мають сформувати комплексну позицію власного проєкту (три ключових)
1.	Низька вартість, зручність у використанні	Визначити нижчу ціну та реалізувати систему, яка буде зручна у використанні та кросплатформенна.	Низька ціна, зручність та зрозумілість інтерфейсу користувача, кросплатформенність.	Швидкість, стабільність, надійність, зручність.

Завдяки дослідженням, проведеним у даному розділі, було створено систему рішень щодо ринкової поведінки компанії, що буде визначати напрям її роботи на ринку:

- цільова група: НТУУ «КПІ ім. Ігоря Сікорського»;
- альтернатива розвитку проєкту: створення автономної системи, що буде функціонувати саме у «КПІ»;
- стратегія охоплення ринку: стратегія концентрованого маркетингу;
- базова стратегія розвитку: стратегія спеціалізації;
- стратегія конкурентної поведінки: стратегія заняття конкурентної ніші.

5.5 Розроблення маркетингової програми стартап-проєкту

Першим кроком є формування маркетингової концепції товару, який отримає споживач (таблиця 5.18). Для цього треба підсумувати результати попереднього аналізу конкурентоспроможності товару.

Таблиця 5.18

Визначення ключових переваг концепції потенційного товару

№ п/п	Потреба	Вигода, яку пропонує продукт	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)
1.	Простота використання	Зрозумілий інтерфейс для простої роботи з системою	Зручний та зрозумілий інтерфейс, створений за найкращими практиками.
2.	Доступність	Кросплатформенність завдяки формату веб-застосунку	Доступність до системи з будь-якого пристрою

Надалі розробляється трирівнева маркетингова модель товару: уточнюється ідея продукту, його складові, особливості його надання (таблиця 5.19).

Таблиця 5.19

Опис трьох рівнів моделі товару

Рівні товару	Сутність та складові		
I. Товар за задумом	Створити систему автоматизації планування навчального навантаження кафедри		
II. Товар у реальному виконанні	Властивості/характеристики	М/Нм	Вр/Тх/Тл/Е/Ор
	1. Надання зрозумілого інтерфейсу користувача, кросплатформеність.	-	-

Продовження таблиці 5.19

	2. Можливість формування форми К-3. 3. Можливість формування плану розподілу навчального-навантаження науково-педагогічних працвників кафедри. 4. Можливість формування індивідуального плану навантаження науково-педагогічного працівника.		
	Якість: стандарти, нормативи, параметри тестування		
	Марка: «Навчальне навантаження КПІ»		
ІІІ. Товар із підкріпленням	До продажу: презентація програмного продукту		
	Після продажу: підтримка клієнтів, розширення функціоналу.		
За рахунок чого потенційний товар буде захищено від копіювання: закритий код і ліцензування.			

Наступним кроком є визначення оптимальної системи збуту (таблиця 5.20), в межах якої приймається рішення:

- проводити збут власними силами або залучати сторонніх посередників;
- вибір та обґрунтування оптимальної глибини каналу збуту;
- вибір та обґрунтування виду посередників.

Таблиця 5.20

Формування системи збуту

№ п/п	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
1.	Клієнти купують програмний продукт безпосередньо у компанії-виробника	Налаштування програмного продукту на матеріально-технічній базі користувача.	Канал нульового рівня (виробник безпосередньо продає товар клієнту).	Через адміністрацію клієнта.

Останньою складовою маркетингової програми є розроблення концепції маркетингових комунікацій, що спирається на попередньо обрану основу для позиціонування, визначену специфіку поведінки клієнтів (таблиця 5.21).

Таблиця 5.21

Концепція маркетингових комунікацій

№ п/п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
1.	Клієнти отримують інформацію	Інтернет, соціальні мережі	Низька ціна, зручність у користуванні	Поширення інформації	Перелік основної

				про оновлення	інформації про

Продовження таблиці 5.21

	про оновлення системи через поштові розсилання				оновлення програмного забезпечення; науково-професійний стиль

Результатом даного розділу є визначення маркетингової програми, що включає в себе концепції товару, збуту, просування, що спирається на цінності та потреби потенційних споживачів, конкурентні переваги ідеї, стан та динаміку ринкового середовища, в якому буде впроваджуватись проєкт, і обрану альтернативу ринкової поведінки.

5.6 Висновки до розділу 5

В результаті проведення маркетингового аналізу стартап-проєкту були визначені загальні напрями використання потенційного програмного продукту, а також його відмінність від конкурентів.

Було визначено наступні показники стартап-проєкту:

- на програмний продукт існує попит, що означає, що комерціалізація проєкту має сенс;
- реалізація та імплементація продукту є доцільною та залишає можливості для вдосконалення;
- конкуренція на ринку є низькою, на відміну від попиту, тому високою є перспектива впровадження продукту за успішної реалізації.

ВИСНОВКИ

В результаті проведеного дослідження було створено систему для автоматизації планування навчального навантаження кафедри.

Перед створенням системи було проаналізовано існуючі програмні рішення, що в певній мірі вирішували дану задачу. До уваги було прийнято функціональність наявних програмних продуктів, переваги та недоліки. Хоча кількість альтернативних варіантів, що доступні для використання, і була великою, в результаті дослідження було виявлено недостатність пропонованого функціоналу існуючого програмного забезпечення у межах поставленої задачі.

Для програмної реалізації системи автоматизації планування навчального навантаження кафедри було обрано актуальні технології розробки, такі як HTML, CSS, JavaScript na його бібліотеки, що дозволило створити надійну, оптимізовану систему, яка є легкою для підтримки та майбутнього розширення функціоналу або масштабування можливостей програмного забезпечення в загальному.

Розроблена система у повному обсязі виконує поставлені до неї завдання, так як вимоги до функціоналу були окремо виділені перед початком розробки потенційними користувачами створеного програмного забезпечення. В результаті система вирішує завдання методистів кафедри з планування навчального навантаження кафедри.

Розроблена система є веб-застосунком, тобто вона є повністю кросплатформенною. Для запуску та роботи з системою користувачеві необхідно мати веб-браузер, та на теперішньому етапі розробки встановлене середовище Node.js.

СПИСОК ВИКОРСТАНИХ ДЖЕРЕЛ

1. Навчальні та робочі навчальні плани [Електронний ресурс] – Режим доступу: https://kpi.ua/regulations-3-2.
2. ПП «Політек-СОФТ» [Електронний ресурс] – Режим доступу: http://www.politek-soft.kiev.ua.
3. Білощицький А.О. Методи та моделі комплексного інформаційно- освітнього середовища в умовах розвитку вищого навчального закладу: Автореф... дис. канд. техн. наук: 05.13.06 — Київ, 2007. — 23с.
4. Науково-дослідний інститут прикладних інформаційних технологій [Електронний ресурс] – Режим доступу: http://www.ndipit.com.ua.
5. АСУ «ВНЗ» [Електронний ресурс] – Режим доступу: http://stservice.com.ua/index.php/29-ask-vnz/81-2016-08-09-07-27-19.
6. Microsoft Excel [Електронний ресурс] – Режим доступу: https://www.microsoft.com/uk-ua/microsoft-365/excel.
7. Using Excel for Statical Data Analysis [електронний ресурс] – Режим доступу: https://people.umass.edu/~evagold/excel.html.
8. Microsoft Access [Електронний ресурс] – Режим доступу: https://www.microsoft.com/ru-ru/microsoft-365/access.
9. Стаття 56 Закону України «Про вищу освіту» [Електронний ресурс] – Режим доступу: https://zakon.rada.gov.ua/laws/show/2145-19#Text.
10. Положення про планування та облік педагогічного навантаження науково-педагогічних працівників КПІ ім. Ігоря Сікорського [Електронний ресурс] – Режим доступу: https://zakon.rada.gov.ua/laws/show/2145-19#Text.
11. Документація Visual Studio Code [Електронний ресурс] – Режим доступу: https://code.visualstudio.com/docs.
12. bin Uzayr S. VS Code Extensions. *Mastering Visual Studio Code*. Boca Raton, 2022. С. 161–230.
13. Callihan S. E. HTML essentials. 2-ге вид. St. Paul, MN : EMC Paradigm, 2010. 468 с.

14. Документація HTML [Електронний ресурс] – Режим доступу: https://developer.mozilla.org/ru/docs/Web/HTML.
15. Brown T. B. CSS Master. SitePoint, 2018. 354 с.
16. Budd A., Björklund E. CSS Mastery. Apress, 2016. 409 с.
17. Документація CSS [Електронний ресурс] – Режим доступу: https://developer.mozilla.org/ru/docs/Learn/Getting_started_with_the_web/CSS_basics.
18. Mayer G. E. T., Awesomeness J. JavaScript: JavaScript Awesomeness Book. CreateSpace Independent Publishing Platform, 2017. 68 с.
19. Preuitt S. Mission JavaScript. Lerner Publishing Group, 2019. 32 с.
20. Timms S., Antani V., Mantyla D. JavaScript: Functional Programming for JavaScript Developers. Packt Publishing, 2016. 646 с.
21. Документація JavaScript [Електронний ресурс] – Режим доступу: https://developer.mozilla.org/ru/docs/Web/JavaScript.
22. Документація Bootstarp [Електронний ресурс] – Режим доступу: https://getbootstrap.com/docs/5.2/getting-started/introduction/.
23. Matsinopoulos P. Practical Bootstrap. Berkeley, CA : Apress, 2020. URL: https://doi.org/10.1007/978-1-4842-6071-5 (дата звернення: 29.11.2022).
24. Mead A. Learning Node.js Development: Learn the fundamentals of Node.js, and deploy and test Node.js applications on the web. Packt Publishing, 2018. 658 с.
25. Syed B. Beginning Node.js. Apress, 2014. 308 с.
26. Wexler J. Get Programming with Node.js. Manning Publications, 2019. 480 с.
27. Документація Node.js [Електронний ресурс] – Режим доступу: https://nodejs.org/uk/docs/.
28. Moroney L. The Definitive Guide to Firebase: Build Android Apps on Google's Mobile Platform. Apress, 2017. 275 с.
29. Singh H., Tanna M. Serverless Web Applications with React and Firebase: Develop real-time applications for web and mobile platforms. Packt Publishing, 2018. 284 с.
30. Документація Firebase [Електронний ресурс] – Режим доступу: https://firebase.google.com/docs.
31. Gourley D., Totty B. HTTP: The Definitive Guide. Tandem Library, 2002.

32. Документація SheetJS [Електронний ресурс] – Режим доступу: https://sheetjs.com.

33. Berson A. Client/server architecture. 2-ге вид. New York : McGraw-Hill, 1996. 569 с.

34. Kolawa A., Huizinga, D. Automated Defect Prevention: Best Practices in Software Management. Wiley-IEEE Computer Society Press, 2007. 75 с.

35. Розроблення стартап-проекту [Електронний ресурс] : Методичні рекомендації до виконання розділу магістерських дисертацій для студентів інженерних спеціальностей / За заг. ред. О.А. Гавриша. – Київ
: НТУУ «КПІ», 2016. – 28 с.

ДОДАТОК А

Інформаційна технологія для автоматизації планування навчального навантаження кафедри

УКР.НТУУ«КПІ ім. Ігоря Сікорського»7186_22М

Аркушів 5

2022

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ
НАВЧАЛЬНО-НАУКОВОГО ІНСТИТУТУ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
XV НАУКОВО-ТЕХНІЧНА КОНФЕРЕНЦІЯ

СУЧАСНІ ІНФОКОМУНІКАЦІЙНІ ТЕХНОЛОГІЇ

9 грудня 2022 року

ЗБІРНИК ТЕЗ

м. Київ

Науково-технічна конференція «Сучасні інфокомунікаційні технології» Збірник тез. К.ДУТ, 2022 – 126с.

Даний збірник містить тези учасників конференції, представлених на XV Науково-технічній конференції студентів та молодих вчених факультету Інформаційних технологій «Сучасні інфокомунікаційні технології», яка проводилась 9 грудня 2022 р. в Навчально-науковому інституті інформаційних технологій Державного університету телекомунікацій, м.Київ.

Робоча мова конференції – українська.

У збірнику представлені тези доповідей XV Науково-технічної конференції студентів та молодих вчених факультету Інформаційних технологій «Сучасні інфокомунікаційні технології». Розглянуті сучасні проблеми розвитку науки і техніки та визначено шляхи їх вирішення.

Вчений секретар конференції
Сава О. І.
моб.тел.+380442492558
e-mail:
Sashasava38@gmail.com

ЗМІСТ

Осипов Владислав Ігорович,
студент 6 курсу, групи ТР-11мп
Національного технічного університету України
«Київський політехнічний інститут імені Ігоря Сікорського»,
(095)1802370,
gektoradonis@gmail.com
Науковий керівник: ***Шушура Олексій Миколайович***
д.т.н., доцент, професор кафедри Цифрових технологій в енергетиці
Національного технічного університету України
«Київський політехнічний інститут імені Ігоря Сікорського», м. Київ

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ДЛЯ АВТОМАТИЗАЦІЇ ПЛАНУВАННЯ НАВЧАЛЬНОГО НАВАНТАЖЕННЯ КАФЕДРИ

Робота викладача пов'язана не тільки з навчанням інших, а й з заповненням великої кількості документації з предметів, які вони викладають. Та на деяких лягає ще серйозніша роль — формування робочого навантаження для кафедри, що базується на навчальному плані. Основною проблемою, з якою стикається працівник — це формат отримуваних даних, роботу з яким досить важко автоматизувати, що змушує перераховувати усі дані самостійно, що призводить до збільшення витраченого часу та ймовірності помилки у розрахунках через людський фактор.

Постановка задачі. Розробка програмного забезпечення, що надасть користувачеві можливість автоматизації процесу планування навчального навантаження з можливістю введення додаткових даних, відсутніх у робочому навчальному плані.

Мета дослідження. Дослідити можливість максимальної автоматизації процесу планування навчального навантаження кафедри з мінімальним втручанням з боку користувача. Розробити програмне забезпечення для реалізації цієї можливості.

Результати дослідження. З розвитком технологій та так званою «діджиталізацією» більшості процесів у сучасному світі комп'ютер стає невід'ємною частиною суспільного життя. Вища освіта також не стоїть на місці — з кожним роком студенти мають отримувати все більше інформації, щоб бути конкурентоспроможними спеціалістами. Цей процес майже повністю залежить від програми навчання, основною складовою якого є навчальний план.

Попередні дослідження демонструють, що створення системи для обчислення педагогічного навантаження кафедри є досить проблематичним. Головною перешкодою для розробки є незручний формат збереження даних в університеті. Проблемою формату є нездатність до його автоматизації. Якщо у таблиці додається хоча б один стовпчик або рядок, то програма почне деякі дані обчислювати абсолютно неправильно. Також не виключається можливість розбіжностей між планами різних навчальних закладів.

Ці фактори спричинюють неможливість створення загальної системи обчислення педагогічного навантаження для вищих навчальних закладів. Якби така система і існувала, то обсяг коду був би величезним, так як програма проходила б велику кількість перевірок на різні формати даних та знаходження

текстових даних. А знайти чисельні дані при зміні формату взагалі неможливо. Все вищезгадане породжує необхідність створення локальної системи для обчислення педагогічного навантаження кафедри.

Розроблена система надасть можливість користувачеві можливість формувати Excel файли різних форматів, підлаштованих під задачі користувача, отримувати навантаження з конкретної дисципліни, закріпляти за нею викладача, перевіряти дані на правильність (мінімальна та максимальна кількість годин для різних викладачів(доценти, аспіранти тощо), кількість студентів у групах тощо), вносити корективи у файли до можливості їх завантажити.

Основною технологією розробки системи було обрано Flutter. Flutter – фреймворк з відкритим кодом для створення мобільних, веб та десктоп додатків, розроблений компанією Google, перша версія якого була запущена у травні 2017 року. Flutter використовує мову програмування Dart, яка також є розробкою Google. Головною перевагою цієї технології є кросплатформеність, що значно спрощує розробку додатків.

Кількість сторонніх бібліотек для Flutter також невпинно зростає. Саме одну з таких, а саме SQFLite, який надає змогу маніпулювати базами даних SQLite, буде використано для створення та роботи з базою даних викладачів.

Для початку роботи з системою користувач повинен завантажити файл з робочим навчальним планом до системи. Після чого, скориставшись функціоналом програми сформувати файл з педагогічним навантаженням кафедри, додавши необхідну додаткову інформацію, що може змінюватись безпосередньо кафедрою (кількість потоків, кількість підгруп у групах, де загальна кількість студентів більша за задане значення). Наступним кроком буде формування навантаження для конкретної дисципліни, після чого потрібно закріпити викладача зі списку за цією дисципліною. Останнім пунктом роботи з системою буде завантаження файлу навантаження для конкретного викладача. Також буде надана можливість відредагувати файл, у випадку, якщо кількість годин навантаження викладача не відповідає його вченому ступеневі тощо.

В результаті проектування було розроблено інформаційну систему для обчислення навчального навантаження кафедри, яка завдяки консультаціям з працівниками кафедри, що працюють з навантаженнями, буде виконувати більшість задач, необхідних для обчислення, чим зменшуватиме вірогідність помилки у розрахунках через людський фактор та значно пришвидшуватиме виконання цього спектру задач.

Висновки та перспективи. Завдяки розробленому програмному забезпеченню час, який витрачається на процес планування навчального навантаження кафедри, та вірогідність помилки під час проведення обчислень значно зменшуються. Перспективою даної технології є впровадження її на кафедрі для подальшого її тестування у робочих умовах, націлених на виявлення недоліків у користуванні та пошук можливостей для розширення її функціоналу.

Список використаних джерел

1. Офіційний сайт фреймворку Flutter [Електронний ресурс]: [Веб-сайт]. – електронні дані. – Режим доступу: https://flutter.dev