КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет комп'ютерних наук та кібернетики
Кафедра інтелектуальних програмних систем

Кваліфікаційна робота
на здобуття ступеня магістра

за спеціальністю 121 Інженерія програмного забезпечення

на тему:

РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ СТВОРЕННЯ ЗВІТІВ ПРО РОЗКЛАД ЗАНЯТЬ

Виконав студент 2-го курсу магістратури
Шумейко Максим Володимирович _____
(підпис)

Науковий керівник:
доцент, кандидат технічних наук
Демківський Євген Олександрович _____
(підпис)

Засвідчую, що в цій роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Роботу розглянуто й допущено до захисту на засіданні кафедри інтелектуальних програмних систем
«__»_____202_р.,
Протокол № __
Завідувач кафедри
Провотар О.І. _____
(підпис)

Київ – 2021

РЕФЕРАТ

Обсяг роботи 50 сторінок, 19 ілюстрацій, 2 таблиці, 15 використаних джерел.

Ключові слова: ВЕБ-СИСТЕМА, ЕЛЕКТРОННА ТАБЛИЦЯ, ЗВІТ, ІНТЕРФЕЙС, РОЗКЛАД, ПРИКЛАДНИЙ ПРОГРАМНИЙ ІНТЕРФЕЙС, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, ШАБЛОН.

Об’єктом розроблення програмного засобу є процес створення звітів про розклад занять.

Метою кваліфікаційної роботи є створення програмного забезпечення для створення звітів про розклад занять.

Результати роботи: визначено типи звітів про розклад та розроблено їх шаблони, досліджено особливості взаємодії з прикладним програмним інтерфейсом веб-сервісу, реалізовано програмний продукт та розгорнуто на сервері. Програмний засіб може застосовуватися навчальними закладами, яким необхідно створювати різного виду звіти про розклад у вигляді електронних таблиць. Його функціональність можна легко розширити для довільної кількості нових шаблонів.

ЗМІСТ

ВСТУП

Оцінка сучасного стану об’єкта розробки. Під час навчального процесу, однією із важливих функцією є складання розкладів. Потрібно враховувати велику кількість факторів, таких як зайнятість викладачів, аудиторій, відсутність будь-яких накладок та багато інших. На сьогодні існує багато програмних засобів, що вирішували б цю задачу. Також є ресурси, що відображають інформацію про заняття у зручному для користувача вигляді.

Актуальність роботи та підстави для її виконання. Розклад занять зазвичай розміщується на стендах у приміщеннях, де вони проводяться, що не досить зручно для більшості студентів та викладачів, оскільки він має великі розміри і пошук необхідної інформації може займати багато часу. Веб-ресурси, які вирішують таку проблему, в той самий час не підходять для самих навчальних закладів, оскільки є певні вимоги до обов’язкових звітів. Тому актуальним є розробка програмних засобів, за допомогою яких можна створювати різноманітного типу звіти про розклад занять, які були б універсальними для навчальних закладів.

Мета й завдання роботи. Метою кваліфікаційної роботи є створення програмного забезпечення для створення звітів про розклад занять. Для досягнення цієї мети поставлено такі завдання:

– визначити типи звітів про розклад та розробити їх шаблони;

– дослідити особливості взаємодії з прикладним програмним інтерфейсом веб-сервісу;

– реалізувати програмний продукт;

– розгорнути програмне забезпечення на сервері.

Об’єкт, методи й засоби розроблення. Об’єктом розроблення програмного засобу є процес створення звітів про розклад занять.

В якості інструменту створення програмного засобу було обрано JetBrains PyCharm 2020.2.2 (Professional Edition) – інтегроване середовище розробки (IDE)

мовою програмування Python. Використано бібліотеки XlsxWriter, Requests – для роботи з електронними таблицями та HTTP-запитами відповідно. Для роботи із веб-складовою використовувався високорівневий відкритий Python-фреймворк Django.

Можливі сфери застосування. Програмний продукт може застосовуватися навчальними закладами, яким необхідно створювати різного виду звіти про розклад у вигляді електронних таблиць. Його функціональність можна легко розширити для довільної кількості нових шаблонів.

РОЗДІЛ 1
ТЕХНОЛОГІЇ РОЗРОБКИ

1.1 Мова програмування Python

Для реалізації програмного забезпечення було обрано мову програмування Python. Це інтерпретована, високорівнева мова програмування, для якої характерна динамічна строга типізація і автоматичне управління пам'яттю. Мова є об'єктно-орієнтованою, оскільки все є об'єктами. Особливістю мови є виділення блоків коду пробільними відступами. Вона підтримує декілька парадигм програмування, серед яких: об’єктно-орієнтовна, аспектно-орієнтовна, функціональна, процедурна та метапрограмування.

Оскільки Python забезпечує динамічну типізацію, тобто лише під час виконання програми визначається тип змінної, то можемо говорити про прив’язування значення до деякого ім'я. Серед типів можна виділити: примітивні (цілочисельний, булевий, число з плаваючою комою, комплексне число) та контейнерні (словник, множина, рядок, список, кортеж). Щоб додати новий тип, потрібно написати клас (class) або визначити новий тип в модулі розширення. Також об’єкти поділяються на змінні та незмінні. До других відносяться списки, множини та словники. Всі інші є незмінні.

Можливості Python часто схиляють до себе розробників під час вибору мови програмування. Серед них можна визначити: модулі й пакети (застосунок або бібліотека оформлюються у вигляді модулей, які можуть збиратись у пакети, а також розміщуватись як у каталогах, так і у вигляді ZIP архівів), підтримку інтроспекції (для довільного об’єкта можна отримати інформацію про його внутрішню структуру), обробку виключень, ітератори, генератори (функції, що зберігають внутрішній стан: поточну інструкцію та значення локальних змінних), декоратори (об’єкти, що приймають в якості аргументу іншу функцію)

та регулярні вирази (є частиною стандартної бібліотеки і їх формат успадкований із Perl) [1].

До переваг цієї мови можна віднести:

– легка у засвоєнні – має мало ключових слів, просту структуру та чітко визначений синтаксис;
– легко читається - код чіткіше визначений і краще сприймається очима;
– легкий в обслуговуванні – вихідний код досить простий в обслуговуванні;
– широка стандартна бібліотека – основна частина бібліотеки Python є дуже портативною та сумісною на різних платформах для UNIX, Windows та Macintosh;
– інтерактивний режим – має підтримку інтерактивного режиму, який дозволяє проводити інтерактивне тестування та налагодження фрагментів коду;
– портативна – може працювати на різноманітних апаратних платформах і має однаковий інтерфейс на всіх платформах.
– розширювана – існує можливість приєднувати модулі низького рівня до інтерпретатора, що дозволяє додавати або налаштовувати свої інструменти, щоб бути ефективнішими;
– бази даних – забезпечує інтерфейси до всіх основних комерційних баз даних;
– програмування графічного інтерфейсу - підтримує графічні інтерфейси, які можна створювати та переносити на багато системних викликів, бібліотек та систем, таких як Windows MFC, Macintosh та система X Window Unix;
– масштабована – забезпечує кращу структуру та підтримку великих програм.

Проте, є і декілька недоліків, про які необхідно зазначити. Це низька швидкодія (як і у більшості інших інтерпретованих мов програмування), неможливість модифікації вбудованих класів, глобальне блокування

інтерпретатора, відсутність статичної типізації (деякі сторонні модулі дозволяють контролювати типи об’єктів) та деякі моменти у синтаксисі та семантиці мови, що викликають плутанину у багатьох розробників.

1.2 Бібліотека XlsxWriter

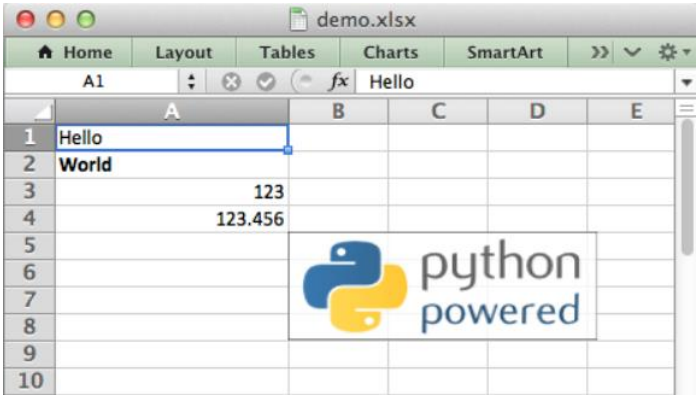
XlsxWriter – це бібліотека Python для запису файлів у форматі Excel 2007+ XLSX [2, 3]. Її можна використовувати для написання тексту, формул, чисел та гіперпосилання на декілька робочих аркушів. Вона підтримує такі функції, як форматування і багато інших, а саме:

– повна сумісність із файлами Excel XLSX;
– повне форматування;
– діаграми;
– об’єднання клітинок;
– визначення імен;
– автоматичні фільтри;
– перевірка даних та випадаючі списки;
– умовне форматування;
– додавання на аркуш PNG/JPEG/BMP/WMF/EMF зображень;
– багатоформатні рядки;
– текстові поля;
– коментарі до комірок;
– підтримка макросів;
– інтеграція з Pandas;
– режим оптимізації пам’яті для запису великих файлів;
– підтримка Python 2.7, 3.4+ і PyPy (використовує лише стандартні бібліотеки).

Розглянемо простий приклад використання бібліотеки (рис. 1.1а), де використано декілька простих функцій, таких як створення файлу, аркушу, додавання тексту, його форматування, вставка зображення тощо.

```python
import xlsxwriter


# Створення нового Excel файлу і додування аркушу
workbook = xlsxwriter.Workbook('demo.xlsx')
worksheet = workbook.add_worksheet()

# Зміна ширини першої колонки.
worksheet.set_column('A:A', 20)

# Додавання жирного шрифта.
bold = workbook.add_format({'bold': True})

# Записати деякий текст.
worksheet.write('A1', 'Hello')

# Форматований текст.
worksheet.write('A2', 'World', bold)

# Запис деяких чисел із використанням формату рядок/колонка.
worksheet.write(2, 0, 123)
worksheet.write(3, 0, 123.456)

# Вставка зображення.
worksheet.insert_image('B5', 'logo.png')

workbook.close()
```

а)



б)

Рис. 1.1. Приклад роботи бібліотеки XlsxWriter: а – лістинінг коду;

б – результат роботи

Функціями, що найчастіше використовуються є об’єднання комірок та багатоформатні тексти. Розглянемо їх детальніше.

Об’єднання виконується за допомогою функції merge_range із наступними параметрами:

– first_row – перший рядок проміжку;

– first_col – перша колонка проміжку;

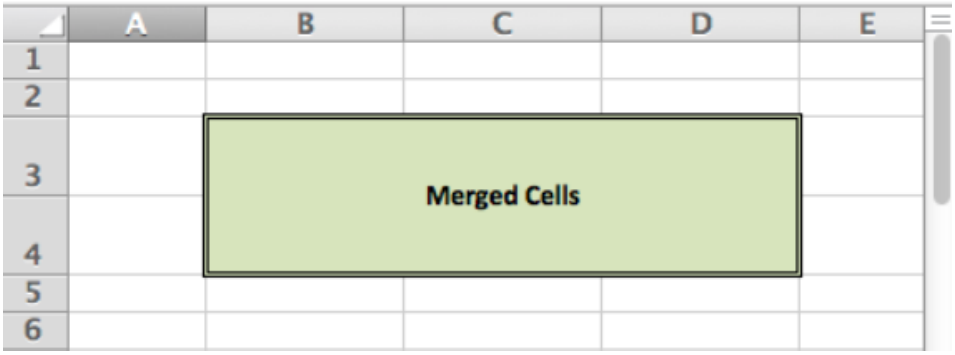
– last_row – останній рядок проміжку;

– last_col – остання колонка проміжку;
– data – значення, яке буде записано після об’єднання;
– cell_format – форматування значення (не обов’язковий).

Наприклад, потрібно об’єднати комірки із проміжку B3:D4 та форматувати їх певним чином. Тоді це можна виконати за допомогою коду зображеного на рис. 1.2 (а) і отримати необхідний результат, як на рис. 1.2 (б).

```
merge_format = workbook.add_format({
    'bold':     True,
    'border':   6,
    'align':    'center',
    'valign':   'vcenter',
    'fg_color': '#D7E4BC',
})

worksheet.merge_range('B3:D4', 'Merged Cells', merge_format)
```

а)



б)

Рис. 1.2. Об’єднання комірок: а – лістинінг коду; б – результат виконання

Для того, щоб записати до комірки текст із декількома форматами, необхідно використати функцію write_rich_string із параметрами:

– row – рядок комірки;
– col – колонка комірки;
– string_parts – пари текстів та їх форматів;
– cell_format – формат комірки (не обов’язковий).

На рис. 1.3 продемонстровано приклад використання вище описаної функції. У ньому частина тексту виділена жирним шрифтом і ще одна курсивом.

```
bold   = workbook.add_format({'bold': True})
italic = workbook.add_format({'italic': True})

worksheet.write_rich_string('A1',
                            'This is ',
                            bold, 'bold',
                            ' and this is ',
                            italic, 'italic')
```

а) б)

Рис. 1.3. Багатоформатний текст: а – лістинінг коду; б – результат роботи

Для уникнення використання тимчасових файлів, при оформленні файлу xlsx, що може знадобитися на серверах, використовується параметр in_memory у конструкторі, значення якого необхідно встановити True.

1.3 Бібліотека Requests

Requests – це бібліотека Python, за допомогою якої можна надсилати всі види HTTP-запитів. Вона проста у використанні та має багато корисних функцій, починаючи від передачі параметрів в URL-адресах, закінчуючи відправкою користувацьких заголовків та перевірки SSL [4].

Ця бібліотека дозволяє надсилати HTTP/1.1 запити. Можна додавати заголовки, дані форм, багатокомпонентні файли та параметри за допомогою простих Python словників і аналогічним чином отримувати доступ до даних відповідей.

HTTP методи, такі як GET і POST, визначають, яку дію ви намагаєтесь виконати під час надсилання HTTP-запиту [5]. Одним з найпоширеніших методів HTTP є GET. Він вказує на те, що ви намагаєтесь отримати дані із зазначеного ресурсу. Щоб зробити запит GET, необхідно викликати метод request.get(). Для перевірки, можна зробити запит GET на Root REST API GitHub, викликавши get () із такою URL-адресою: requests.get('https://api.github.com').

Response (відповідь) – це потужний об’єкт для перевірки результатів запитів. Наприклад, перший біт інформації, який з нього можна отримати – код стану, він повідомляє про стан запиту. Наприклад, статус 200 OK означає, що запит був успішним, а 400 NOT FOUND – що, ресурс не знайдено. Існує велика кількість інших можливих кодів стану, що надають інформацію про результат запиту. Звернувшись до .status_code можна отримати код стану, який повернув сервер: requests.get('https://api.github.com').status_code.

У відповідь на GET запит у тілі повідомлення часто міститься деяка цінна інформація (корисне навантаження). Переглянути її можна, використовуючи методи та атрибути Response:

– .content – переглянути вміст у байтах;
– .text – переглянути вміст у вигляді тексту;
– .json – відображення вмісту у форматі “JSON” (повертає словник, тож значення можна отримувати за ключем).

Поширений спосіб налаштування GET запитів – це передача значень через параметри рядка запиту в URL-адресі. Щоб це зробити за допомогою get(), треба передати необхідні дані в параметри. Наприклад, можна використати API пошуку GitHub, щоб відфільтрувати репозиторії (рис. 1.4).

```
import requests


response = requests.get(
    'https://api.github.com/search/repositories',
    params={'q': 'requests+language:python'},
)


json_response = response.json()
repository = json_response['items'][0]
print(f'Repository name: {repository["name"]}')
print(f'Repository description: {repository["description"]}')
```

Рис. 1.4. Використання параметризованого запиту

Передаючи словник {'q': 'request + language: python'} параметру params у метод .get (), можна змінити результати, що повертаються з API пошуку.

Окрім GET, до інших популярних методів HTTP належать POST, PUT, DELETE, HEAD, PATCH та OPTIONS. Requests надає метод із подібною сигнатурою до get () для кожного з цих HTTP методів:

- requests.post(' site ', data={'key':'value'})
- requests.put(' site ', data={'key':'value'})
- requests.delete(' site/delete ')
- requests.head(' site/get ')
- requests.patch(' site/patch ', data={'key':'value'})
- requests.options(' site/get ')

РОЗДІЛ 2
РОЗРОБКА ШАБЛОНІВ ЗВІТНОЇ ДОКУМЕНТАЦІЇ ПРО ГРАФІК НАВЧАЛЬНОГО ПРОЦЕСУ ТА СЕСІЇ

2.1 Розклад групи

Шаблони із розкладом груп (рис. 2.1) створюються із дотримання наступних вимог:

– шапка для затвердження деканом розташована по праву сторону документу, шрифт – Times New Roman, розмір 14 пунктів, для всієї іншої інформації – розмір 12 пунктів;

– заголовок розкладу повинен бути відцентрований, факультет та спеціальність повинні виділятись жирним шрифтом;

– таблиця складається із заголовків колонок: «День», «Час», «Дисципліна, викладач», «Група», «Тижні», «Аудиторія»; всі назви відцентровані;

– дні повинні розташовуватись вертикально, виділятись жирним шрифтом, та об’єднувати всі рядки із предметами цього дня;

– для кожного дня у колонці «Час» повинно бути кількість рядків, що дорівнює кількості предметів на факультеті, навіть якщо у цей день у цей час немає пари, а також об’єднувати всі рядки із парами у цей час; всі записи відцентровані;

– пари записані у форматі [Назва предмету], [Викладач]; назва предмету виділена жирним шрифтом, а посада викладача та його ПІБ – курсивним;

– колонка «Група» містить номер групи, якщо це семінар або позначення іншого типу пари; записи відцентровані;

– колонка «Тижні» повинна містити номери та/або інтервали тижнів, у які цей предмет викладається; записи відцентровані;

– колонка «Аудиторія» містить номери корпусу та аудиторії через дефіс; записи відцентровані.

Декан факультету _____
підпис прізвище, ім`я, по батькові
"____"_____20_____р.

Факультет інформатики
Спеціальність **"Комп`ютерні науки (бакалаврат)", 1р.н**
Розклад занять на весняний семестр 2019 - 2020 н.р.

День	Час	Дисципліна, викладач	Група	Тижні	Аудиторія
Понеділок	08:30-09:50				
	10:00-11:20	**Англійська мова ,** *ст. викл. О.П. Качурець*	1	1-14	3-403а
		Англійська мова , *ст. викл. Н.В. Соломашенко*	2	1-14	3-408
		Англійська мова , *ст. викл. Т.П. Сорокіна*	3	1-14	3-309
		Англійська мова , *проф. О.Ю. Моісєєнко*	4	1-14	3-302
		Англійська мова , *ст. викл. С.О. Гісем*	5	1-14	8-4
		Англійська мова , *ст. викл. Л.Л. Бачинська*	6	1-14	8-8
		Англійська мова , *ст. викл. Н.М. Максимчук*	7	1-14	3-220а
		Англійська мова , *ст. викл. О.М. Демидович*	8	1-14	8-11
		Англійська мова , *ст. викл. П.Г. Покотило*	9	1-14	2-406
	11:40-13:00	**Алгоритми та структури даних,** *проф. М.М. Глибовець*	лекція	1-7,10-15	1-223
	13:30-14:50				
	15:00-16:20				
	16:30-17:50				
	08:30-09:50				
	10:00-11:20	**Алгоритми та структури даних,** *ас. К.В. Салата*	1	2-7,10-15	1-309
		Дискретна математика, *доц. О.С. Пилявська*	лекція	1	1-225

Рис. 2.1. Приклад звіту розкладу групи

2.2 Розклад факультету

Вимоги до шаблону із розкладом факультету (рис. 2.2):

– шапка для затвердження деканом розташована по ліву сторону документу, шрифт – Times New Roman, розмір 12 пунктів;

– для всієї іншої інформації шрифт шрифт – Calibri;

– документ повинен бути розбитий на блоки по курсам, дням та часом проведення заняття;

– таблиця складається із заголовків колонок: [Курс] (об’єднує всі колонки груп даного курсу, розмір 36 пунктів), [Група] (розмір 24 пункти), [Спеціальність] (об’єднує колонки груп цієї спеціальності, розмір 16 пунктів); всі назви відцентровані, виділенні жирним шрифтом;

– дні повинні розташовуватись вертикально, виділятись жирним шрифтом, та об’єднувати всі рядки із часом проведення пар;

– для кожного часу проведення пари повинно бути чотири рядки, по два на верхній та нижній тижні відповідно, всі записи відцентровані;

– якщо заняття проводиться по верхніх тижнях, то запис повинен об’єднувати два рядки у відповідній колонці, якщо у нижній, то об’єднує два останніх рядки, виділені під кожне заняття; якщо таких обмежень немає, то об’єднуються усі чотири рядки;

– запис предмета повинен містити: назву (виділену жирним шрифтом), тип заняття (виділений жирним шрифтом), номери тижнів проведення, номер підгрупи, корпус та аудиторію проведення (також жирним шрифтом), ступінь та ПІБ викладача;

– якщо занять у групи в якийсь день немає, то позначається як «Самостійна робота», вертикально розміщено та об’єднує усі рядки в колонці цього дня; записи відцентровано, розмір шрифту: 80, виділено жирним шрифтом;

– заняття, які проводяться одночасно для декількох груп повинні об’єднуватись у звіті.

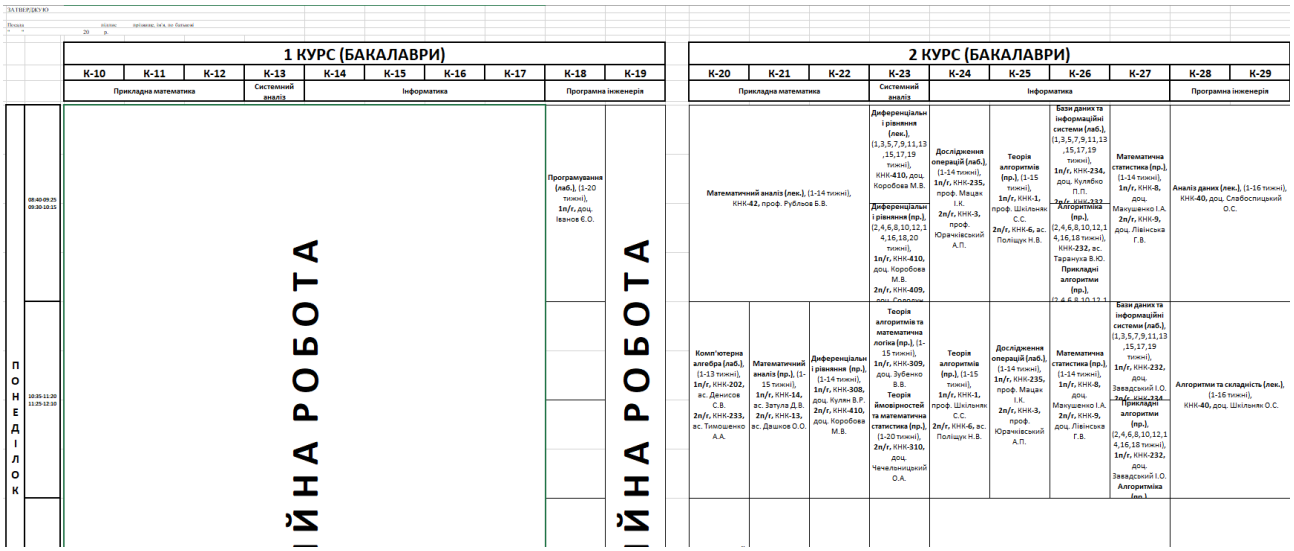


Рис. 2.2. Приклад звіту розкладу факультету

2.3 Розклад аудиторій факультету

Особливості звіту розкладу аудиторій факультету (рис. 2.3):

– шапка звіту розташована по центру документа, виділена жирним шрифтом, розмір: 14 пунктів, шрифт: Calibri;

– весь інший документ оформлений шрифтом Calibri із розміром 11;

– таблиця складається із колонок «№» (номер заняття), «Час» (час проведення заняття), «Ауд.» (корпус та аудиторія проведення заняття) та шість колонок з «Понеділок» по «Субота» із підколонками «Предмет» (назва предметів та викладачі) і «№ т.» (номери тижнів проведення відповідних занять);

– колонки «№» і «Час» повинні об’єднувати усі рядки, у яких присутні записи відповідних занять в цей час;

– колонка «Час» відображається вертикально;

– клітинка, в якій записується аудиторія із предметами, які проводяться у ній в один і той самий час, повинна підсвічуватись жовтим, що буде позначати даний тип накладок;

– клітинка, в якій записується аудиторія із викладачами, хоча б один із яких присутній в цей же час у іншій аудиторії, повинна підсвічуватись червоним, що буде позначати даний тип накладок.

Зайнятість аудиторій
ф-ту Комп'ютерних наук та кібернетики
на 2 семестр 2019-2020 навчального року

№	Час	Ауд.	Понеділок		Вівторок		Середа		Четвер		П'ятниця		Субота	
			Предмет	№ т.	Предмет	№ т.	Предмет	№ т.	Предмет	№ т.	Предмет	№ т.	Предмет	№ т.
		КНК-202	СП, Русіна Н.Г.	1,3,5,7,9,11,13					ООП, Денисов С.В./ ТМКМ, Стоян В.А.	2,4,6,8,10,12,14,16,18,20/	ІСТП, Омельчук Л.Л.	1-19		
		КНК-203	СП, Волохов В.М.	2,4,6,8,10,12,14			ПТП, Кузенко В.Ф.	1,3,5,7,9,11,13,15	КМ, Демківський Є.О./ ООП, Оноцький В.В.	1,3,5,7/ 2,4,6,8,10,12,14,16,18,20	БДІС, Федорус О.М./ А, Тарануха В.Ю.	1,3,5,7,9,11,13,15,17,19/ 2,4,6,8,10,12,14,1		
		КНК-204			ЗДГ, Катеринич Л.О.	1-9			ОООП, Терлецький Д.О.	1-14				
		КНК-205	ІС, Федорус О.М.	1-9	АПДМ, Криволап А.В.	2,4,6,8,10,1	ІС, Федорус О.М.	1-9	КГ, Стовба В.О.	1-13				
		КНК-221	ООП, Галкін О.В.	1-15	ОРЗ, Матвієнко В.Т.	1-7	ФП, Галкін О.В.	1-10	КМ, Демківський Є.О./ ООП, Денисов С.В.	2,4,6,8,10/ 1,3,5,7,9,11,13,15,17,19	ОК, Галкін О.В./ КГ, Шкільняк О.С.	5-20/ 1-4		
		КНК-222							ТПР, Махно М.Ф.	1,3,5,7,9,11,13	ІСТП, Русіна Н.Г.	1-19		
		КНК-231			МОЗІ, Кривий С.Л.	1-9			РО, Верес М.М.	1-14				
		КНК-232	ПА, Завадський І.О./ БДІС, Завадський	2,4,6,8,10,12,14,16,18/ 1,3,5,7,9,11,13,15			Прог, Жереб К.А.	1,3,5,7,9,11,13,15	ЕЕПМ, Колянова Т.В.	2,4,6,8,10,12,14				

Рис. 2.3. Приклад звіту розкладу аудиторій факультету

2.4 Розклад сесії курсу

Шаблон даного типу звіту (рис. 2.4) формується із урахуванням наступних пунктів:

– документ складається із трьох аркушів: «Заліки», «Екзамени», «Інші види контролю» (здача курсових робіт, захист дипломних робіт, перескладання, комісії тощо);

– шапка звіту розташована по праву сторону документа, розмір: 12 пунктів, шрифт: Times New Roman;

– весь інший текст звіту оформлений шрифтом Times New Roman із розміром 11 пунктів;

– аркуш «Заліки» містить наступні колонки: «Група (підгрупа)», «Дисципліна», «Викладач(і)», «Консультації» (підзаголовки: «Дата», «Год.», «Ауд.»);

– аркуш «Інші види контролю» містить аналогічні колонки, як і попередній пункт;

– аркуш «Екзамени» містить колонки попередніх аркушів із додатковою «Екзамени» (підзаголовки: «Дата», «Год.», «Ауд.»);

– записи сортуються по групах(підгрупах), і датах проведення контролю;

– колонка «Група (підгрупа)» повинна об’єднувати рядки із всіма дисциплінами, з яких проводиться контроль для певної групи.

2.5 Розклад викладача

Звіт із розкладом викладача (рис. 2.5) формується із дотриманням наступних вимог:

– весь текст документу оформлений шрифтом Times New Roman розміру 12 пунктів;

– шапка звіту розташована по праву сторону документа;

ЗАТВЕРДЖУЮ
Декан факультету_____
підпис
"_____"_____ 20___Р.

Факультет комп`ютерних наук та кібернетики
Графік залікової сесії на І семестр, 2019-2020 н.р.
студентів 2-го курсу ОКР «Бакалавр»

Група (підгрупа)	Дисципліна	Викладач(і)	Консультації		
			Дата	Год.	Ауд.
К-20	Теорія ймовірностей	ас.Макушенко І.А., доц.Шарапов М.М.	07.12.2020	10:00	КНК-10
К-21 (1)	Математичний аналіз	ас.Макушенко І.А.	09.12.2020	10:00	КНК-231
К-21 (2)	Математичний аналіз	ас.Макушенко І.А.	09.12.2020	12:00	КНК-231
К-27	Математична логіка	доц.Шарапов М.М.	10.12.2020	10:00	КНК-10
	Архітектура обчислювальних систем та комп'ютерні мережі	доц.Шарапов М.М.	11.12.2020	10:00	КНК-705
	Українська та зарубіжна культура	доц.Петренко М.М.	12.12.2020	13:00	Червоний-105

а)

ЗАТВЕРДЖУЮ
Декан факультету_____
підпис
"_____"_____ 20___Р.

Факультет комп`ютерних наук та кібернетики
Графік екзаменаційної сесії на І семестр, 2019-2020 н.р.
студентів 2-го курсу ОКР «Бакалавр»

Група (підгрупа)	Дисципліна	Викладач(і)	Консультації			Екзамени		
			Дата	Год.	Ауд.	Дата	Год.	Ауд.
К-21	Вступ до дослідження операцій	доц. Якимів Р.Я., ас. Заворотинський А.В.	11.12.2020	10:00	КНК-304	14.12.2020	9:00	КНК-304
	Об'єктно-орієнтоване програмування	ас.Голубєва К.М., проф.Клюшин Д.А.	15.12.2020	10:00	КНК-40	16.12.2020	9:00	КНК-40
К-22 (А1)	Іноземна мова	доц. Шатирко А.В., ас. Волошук С.Д.	11.12.2020	10:00	КНК-304	14.12.2020	9:00	КНК-304
К-22 (А2)	Іноземна мова	доц. Шатирко А.В., ас. Волошук С.Д.	11.12.2020	11:00	ФРЕКС-304	14.12.2020	10:00	ФРЕКС-304
К-22 (А2)	Іноземна мова	доц. Шатирко А.В., ас. Волошук С.Д.	11.12.2020	12:00	ФРЕКС-304	14.12.2020	11:00	ФРЕКС-304
К-23	Диференціальні рівняння	доц.Кулян В.Р., доц.Коробова М.В.	11.12.2020	10:00	ФРЕКС-409	14.12.2020	9:00	ФРЕКС-409
	Архітектура обчислювальних систе	доц.Кулян В.Р., доц.Коробова М.В.	11.12.2020	10:00	ФРЕКС-409	14.12.2020	9:00	ФРЕКС-409
	Математичний аналіз-2	доц.Гончаренко Ю.В., ас.Затула Д.В.	21.12.2020	10:00	ФРЕКС-306	22.12.2020	9:00	ФРЕКС-306

б)

Рис. 2.4. Приклад звіту розкладу сесії курсу: а – аркуш «Заліки»; б – аркуш «Екзамени»

– таблиця складається із колонок «№ пари (час)» та «Понеділок»-«Субота»;

– колонка «№пари (час)» повинна відображати усі часові проміжки проведення пари, що відповідають час початку і закінчення пари на відповідних факультетах;

– записи предметів містять назву, виділену жирним шрифтом, формат заняття, групи та аудиторію проведення.

ЗАТВЕРДЖУЮ
Декан факультету_____
підпис
"____"_____20___Р.

РОЗКЛАД ВИКЛАДАЧА
доцент Анікушин Андрій Валерійович
на 2-й семестр 2019/2020 навчального року

№ пари (час)	Понеділок	Вівторок	Середа	Четвер	П'ятниця	Субота
1 (8:30-9:50)	**Математичний аналіз** (лек.), К-17, К-18, К-19, (1-14 тижні), КНК-39	**Алгебра та геометрія (пр.)**, (1-14 тижні), ІПЗ-31, 1 п/г, КНК-309				
2 (10:00-11:20)	Математичний аналіз (пр.), (1,3,5,7,9,11,13,15,17 тижні), КНК-605				Математичний аналіз (пр.), (1-14 тижні), ПЗС-1, КНК-605	
3 (11:20-12:20) (11:40-13:00)		Алгебра та геометрія (пр.), (1-14 тижні), К-22, 2 п/г, ІЖ-309		Алгебра та геометрія (пр.), (1-14 тижні), К-22, 2 п/г, КНК-309		

Рис. 2.5. Приклад звіту розкладу викладача

2.6 Розклад викладачів, що читають на спеціальності

Шаблон (рис. 2.6) схожий до попереднього із деякими доповненнями:

– додається попереду колонка «Викладачі»;

– викладач записується із вказанням наукового ступеню, ПІБ виділяється жирним шрифтом;

– якщо у викладача відбувається накладка (дві пари одночасно), то таку клітинку треба виділяти червоний кольором;

Викладачі	№ пари (час)	Понеділок	Вівторок	Середа	Четвер	П'ятниця	Субота
Доцент **Анікушин Андрій Валерійович**	1 (8:30-9:50)	**Математичний аналіз** (лек.), К-17, К-18, К-19, (1-14 тижні), КНК-39	**Алгебра та геометрія (пр.)**, (1-14 тижні), ІПЗ-31, 1 п/г, КНК-309				
	2 (10:00-11:20)	**Математичний аналіз** (пр.), (1,3,5,7,9,11,13,15,17 тижні), КНК-605				**Математичний аналіз** (пр.), (1-14 тижні), ПЗС-1, КНК-605	
	3 (11:40-13:00)				**Алгебра та геометрія** (пр.), (1-14 тижні), К-22, 2 п/г, КНК-309		
Професор **Провотар Олександ Іванович**	1 (8:30-9:50)						
	2 (10:00-11:20)		**Алгебра та геометрія** (пр.), (1-14 тижні), К-11, 1 п/г, КНК-309				
	3 (11:40-13:00)				**Математичний аналіз** (лек.), (2,4,6,8,10 тижні), ІПЗ-41, ІПЗ-42, КНК-01/**Алгебра та геометрія** (пр.), (1-14 тижні), К-21, 1 п/г, КНК-309		
	4 (13:30-14:50)		**Математичний аналіз** (лек.), (1-3, 7-9, 11				

Рис. 2.6. Приклад звіту розкладу викладачів, що читають на спеціальності

2.7 Розклад сесії викладача

Ще один звіт, який стосується викладача – це розклад його сесії. Створено наступні вимоги до шаблону (рис. 2.7):

– весь текст документу оформлений шрифтом Times New Roman розміру 12 пунктів;

– шапка звіту розташована по праву сторону документа;

– таблиця складається із колонок «День», «Час», «Дисципліна», «Вид контролю», «Група (Підгрупа)», «Дата», «Аудиторія»;

– дні повинні розташовуватись вертикально та об’єднувати всі рядки із часом проведення контролю в цей день;

– дисципліни виділяються жирним шрифтом, вид контролю – курсивом;

– записи повинні бути посортовані за часом та датою проведення.

ЗАТВЕРДЖУЮ
Декан факультету_____
підпис
"_____"_____ 20___Р.

РОЗКЛАД СЕСІЇ ВИКЛАДАЧА
доцент Петренко Петро Петрович
у 2-му семестрі 2019/2020 навчального року

	Час	Дисципліна	Вид контролю	Група (Підгрупа)	Дата	Аудиторія
Понеділок	8:30	**Математичний аналіз**	*консультація*	К-10, К-11, К-12	14.12.2020	3-220а
	10:00	**Математичний аналіз**	*екзамен*	К-10 (1)	14.12.2020	3-220а
	13:00	**Математичний аналіз**	*екзамен*	К-12 (А2)	15.12.2020	3-302
	13:30	**Алгебра**	*залік*	ІПЗ-1	16.12.2020	3-302
	15:00	**Українська мова (за професійним спрямуванням)**	*залік*	ТК-2	16.12.2020	3-302
Вівторок	10:00	**Математичний аналіз**	*консультація*	Фінанси-1	20.12.2020	3-220а
		Алгебра	*залік*	БІТ-1-21 (1)	11.12.2020	3-302
	11:40	**Диференціальні рівняння**	*залік*	2	21.12.2020	1-112

Рис. 2.7. Приклад звіту розкладу сесії викладача

2.8 Інші типи шаблонів

Окрім вище наведених шаблонів, також було реалізовано наступні типи:

– розклад сесії групи;
– розклад курсу по факультету;
– розклад спеціальності;
– розклад аудиторій корпусу;
– розклад аудиторій університету.

Дані шаблони використовують уже описані макети або є їхніми складовими. Звіт «Розклад сесії групи» (рис. 2.8) схожий до «Розклад групи», але із колонками «День», «Час», «Дисципліна, викладач», «Вид контролю», «Група», «Дата», «Аудиторія». «Розклад курсу по факультету» та «Розклад спеціальності» - це частинки макету розкладу факультету, а розклади аудиторій університету та корпусу використовують той же шаблон, що й розклад аудиторій факультету (з виділенням накладок).

День	Час	Дисципліна, викладач	Вид контролю	Група	Дата	Аудиторія
Понеділок	8:30-9:50					
	10:00-11:20	**Лінійна алгебра та аналітична геометрія,** *ст.викл. Л.М. Тимошкевич*	*консультація*	1	14.12.2020	Дистанційно
		Лінійна алгебра та аналітична геометрія, *ас. В.А. Ольшевська*	*консультація*	2	14.12.2020	Дистанційно
		Мови програмування, *ст.викл. О.В. Кирієнко*	*консультація*	1	21.12.2020	Дистанційно
		Мови програмування, *ст. викл. О.М. Пєчкурова*	*консультація*	2	21.12.2020	Дистанційно
	11:40-13:00	**Лінійна алгебра та аналітична геометрія,** *ас. В.А. Ольшевська*	*консультація*	3	14.12.2020	Дистанційно
		Мови програмування, *ст.викл. О.В. Кирієнко*	*консультація*	3	21.12.2020	Дистанційно
		Мови програмування, *ст. викл. О.М. Пєчкурова*	*консультація*	4	21.12.2020	Дистанційно
	13:30-14:50	**Мови програмування,** *ст.викл. О.В. Кирієнко*	*консультація*	5	21.12.2020	Дистанційно
		Мови програмування, *ст. викл. О.М. Пєчкурова*	*консультація*	6	21.12.2020	Дистанційно
	15:00-16:20					
	16:30-17:50					

Рис. 2.8. Приклад звіту розкладу сесії групи

РОЗДІЛ 3
ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ АВТОМАТИЧНОГО СТВОРЕННЯ ЗВІТІВ ПРО ГРАФІК НАВЧАЛЬНОГО ПРОЦЕСУ ТА СЕСІЇ

Оскільки при розробці системи був використаний фрейморк Django [6], то вона має архітектуру MVT (Model – View – Template) [7], схема якої зображена на рис. 3.1 та відповідає архітектурі MVC (Model – View - Controller).

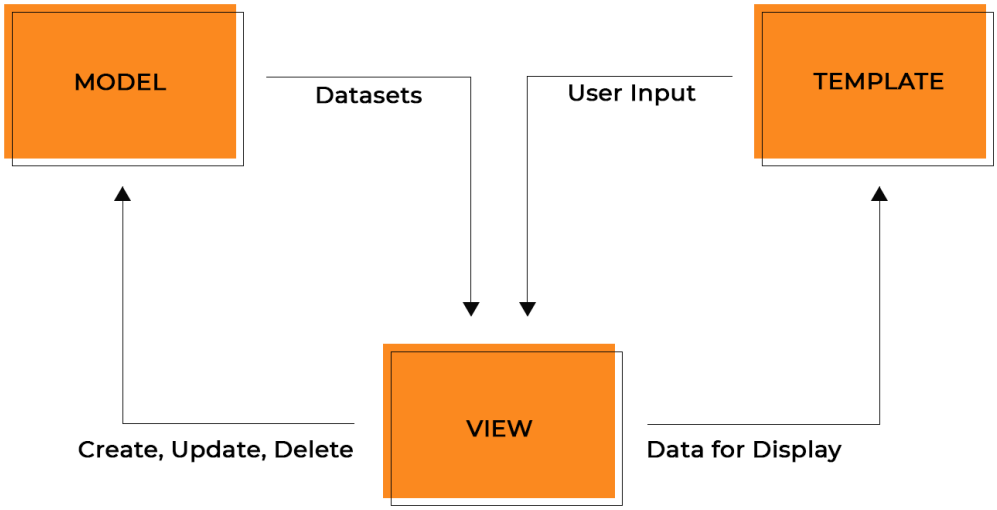


Рис. 3.1. Схема архітектури системи

Моделі – класи, які містять інформацію про дані. Ці дані представлені атрибутами та полями. Зазвичай одна модель відповідає одній таблиці в базі даних.

У розробленій системі використані наступні моделі з відповідними полями:

– університет (назва, коротка назва, поточний семестр);
– факультет (назва, коротка назва, університет, корпуси);
– спеціальність (назва, коротка назва, факультет);
– курс (назва, кваліфікаційний рівень, спеціальність);
– група (назва, коротка назва, підгрупи, курс, рік, семестр);

– спеціальність (назва, коротка назва, факультет);

– рік (початок, кінець, університет);

– семестр (номер, рік, початок, кінець);

– спеціальність (назва, коротка назва, факультет);

– пара (назва предмету, корпус, аудиторія, час пари, формат, день, підгрупа, тижні, спосіб викладання, викладачі);

– контроль (назва предмету, тип, корпус, аудиторія, час, дата, формат проведення, підгрупа, викладачі);

– час пари (початок, кінець, номер, наявність перерви, початок перерви, кінець перерви, факультет);

– корпус (назва, коротка назва, адреса, локація, поверхи, університет);

– аудиторія (номер, поверх, корпус);

– викладач (прізвище, ім’я, по-батькові, ступінь).

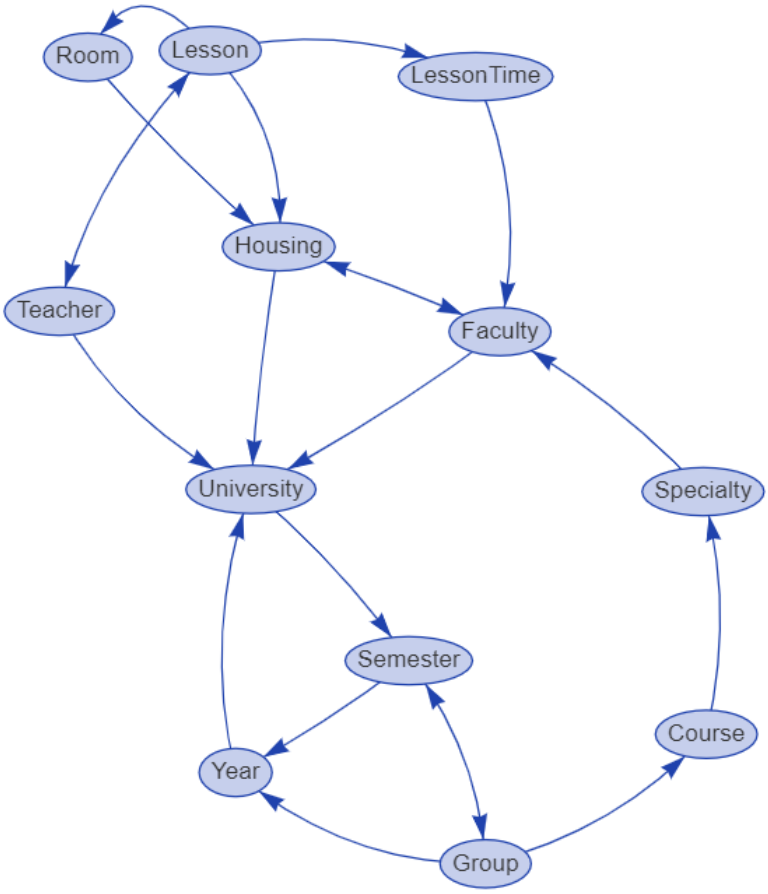


Рис. 3.2. Схема відношень моделей системи

Представлення (view) виконує три функції [8]: приймає HTTP-запити, реалізує бізнес-логіку, яка визначається методами і властивостями та відправляє HTTP-відповіді на отримані запити.

У програмі за це відповідає окремий модуль. З веб-форми він отримує всю введену користувачем інформацію, таку як університет, тип звіту та додаткові параметри, в залежності від обраного типу. Далі отримані дані обробляються відповідно до необхідного звіту:

а) розклад групи;

– в базі відбувається її пошук;

– робиться запит предметів із фільтрацією по групі та семестру (задається поточний по замовчуванню);

– одержані результати сортуються за днем, часом початку заняття та підгрупою;

– формується файл із звітом відповідно до шаблону;

– збереження файлу із наступною назвою: [Коротка назва університету]-[Коротка назва факультету]-[Номер семестру]-[Навчальний рік]-[Назва групи];

б) розклад факультету;

– в базі відбувається його пошук;

– робиться запит всіх спеціальностей, що відповідають поточному семестру та освітньо-кваліфікаційному рівню;

– вибираються групи, які належать до одержаних спеціальностей та сортуються за курсом;

– для них запитуються усі предмети поточного семестру, групуються та сортуються за днем і часом початку заняття;

– формується файл із звітом відповідно до шаблону;

– збереження файлу із наступною назвою: [Коротка назва університету]-[Коротка назва факультету]-[Номер семестру]-[Навчальний рік]-[Освітньо-кваліфікаційний рівень];

в) розклад аудиторій факультету;

– в базі відбувається пошук факультету;

– робиться запит всіх спеціальностей, що відповідають поточному семестру;

– вибираються групи, які належать до одержаних спеціальностей;

– для них запитуються усі предмети поточного семестру;

– з одержаних результатів створюється список аудиторій для яких запам’ятовуються викладачі та предмети, відповідно до часу та дня проведення;

– знаходяться усі накладки для аудиторій і помічаються відповідно до їх типів (два викладачі знаходяться в аудиторії в один і той же час, викладач в один і той же час повинен викладати в різних аудиторіях);

– формується файл із звітом відповідно до шаблону;

– збереження файлу із наступною назвою: «Зайнятість-аудиторій»-[Коротка назва університету]-[Коротка назва факультету]-[Номер семестру]-[Навчальний рік];

г) розклад сесії для групи;

– в базі відбувається пошук групи;

– робиться запит усіх предметів, для яких в поточному семестрі проводиться контроль;

– одержані результати сортуються за днем, часом проведення та датою;

– формується файл із звітом відповідно до шаблону;

– збереження файлу із наступною назвою: «Сесія»-[Коротка назва університету]-[Коротка назва факультету]-[Назва групи]-[Освітньо-кваліфікаційний рівень]-[Номер семестру]-[Навчальний рік];

г) розклад сесії для курсу;

– в базі знаходиться заданий курс;

– для курсу робляться запити для пошуку спеціальностей цього курсу, а потім усіх груп, що відповідають курсу та спеціальностям;

– для кожної групи та їх підгруп робиться запит із пошуком заліків для них у поточному семестрі;

– у аркуші «Заліки» результуючого файлу записуються одержані дані, відповідно до шаблону, де записи сортуються за групами та датами проведення контролю;

– для кожної групи та їх підгруп робиться запит із пошуком екзаменів для них у поточному семестрі;

– у аркуші «Екзамени» результуючого файлу записуються одержані дані, відповідно до шаблону, де записи відсортовані за групами та датами проведення консультацій та екзаменів;

– аналогічні попереднім двом пунктам дії робляться і для інших видів контролю, результати записуються в аркуш «Інші види контролю»;

– збереження файлу із наступною назвою: «Сесія»-[Коротка назва університету]-[Коротка назва факультету]-[Номер курсу]-[Освітньо-кваліфікаційний рівень]-[Номер семестру]-[Навчальний рік];

д) розклад викладача;

– в базі проводиться пошук введеного викладача;

– проводиться запит із предметами, які ним викладаються у поточному семестрі;

– одержані результати сортуються і групуються за днем та часом проведення;

– формується файл із звітом відповідно до шаблону;

– збереження файлу із наступною назвою: «Розклад»-[ПІБ у короткій формі]-[Коротка назва університету]-[Коротка назва факультету]-[Номер семестру]-[Навчальний рік];

Для інших типів звітності, дані обробляються схожим чином до вище описаних, враховуючи ключову модель для пошуку. Наприклад, розклад аудиторій корпусу та університету обробляються так само, як і для факультету, тільки при фільтрації предметів, що викладаються групам будуть враховуватись аудиторії які відповідають введеному корпусу та університету відповідно.

Сформований файл відправляється користувачеві в якості HTTP-відповіді.

Представлення надає дані та отримує їх з шаблонів (templates). Вони представляються собою файли з HTML-кодом, за допомогою якого відображаються дані, та не містять бізнес-логіки.

У реалізованих шаблонах було розроблено користувацький інтерфейс та Java Script функції, які дозволяють скинути усі вибрані на формі параметри, а також відслідковує, щоб користувач не зміг завантажити звіт, якщо не були обрані усі необхідні для цього дані. Ще одна важлива функція – це заповнення даними усіх випадаючих списків. Вона відстежує вибрані поля та додає необхідну інформацію, взяту за запитом із бази даних, до пов’язаних полів. Спочатку заповнюється список із університетами. Якщо обрано тип звіту «Розклад групи», то послідовно будуть оновлюватись наступні поля: «Факультет», «Спеціальність» та «Група». Якщо користувач вибрав «Розклад факультету», то – «Факультет», «ОКР».

Ще однією реалізованою функціональністю є створення посилань, за якими можна отримати звіт із адресами, які ведуть на веб-ресурс mytimetable.live із відповідним розкладом. Такими є посилання на:

– розклад занять факультету;
– розклад занять викладачів;
– розклад занять спеціальності;
– розклад сесії факультету;
– розклад сесії спеціальності;
– розклад сесії викладачів факультету.

Список таких посилань формується за шаблоном: «[Іконка] + [Посилання]». Посилання складається із категорії розкладу (для груп,

викладачів, їх сесії) та короткої назви англійською (групи або ПІБ викладача). Приклад зображено на рис. 3.3.

Рис. 3.3. Приклад посилань на розклад занять факультету

Списки розробляються відповідно до категорій наступним чином:

а) розклад занять факультету;

– в базі знаходиться факультет;

– робиться запит на курси цього факультету із розбивкою на ОКР;

– формується список для даної категорії;

б) розклад занять спеціальності (аналогічно до факультету);

в) розклад занять викладачів;

– в базі знаходиться факультет;

– формується відповідне посилання;

г) розклад сесії факультету (аналогічно до розкладу занять, тільки посилання формуються відповідно із категорією сесії);

г) розклад сесії спеціальності (аналогічно розкладу сесії факультету);

д) розклад сесії викладачів факультету (аналогічно заняттям; посилання з категорією сесії).

РОЗДІЛ 4
ОСОБЛИВОСТІ ВЗАЄМОДІЇ З ПРИКЛАДНИМ ПРОГРАМНИМ ІНТЕРФЕЙСОМ ВЕБ-СЕРВІСУ

4.1 Поняття прикладного програмного інтерфейсу

Прикладний програмний інтерфейс (API) дозволяє компаніям відкривати дані та функції своїх програм для зовнішніх сторонніх розробників, ділових партнерів та внутрішніх підрозділів у своїх компаніях [9]. Це дозволяє послугам та продуктам взаємодіяти між собою та використовувати дані та функціональність один одного за допомогою задокументованого інтерфейсу. Розробникам не потрібно знати, як реалізований API, вони просто використовують інтерфейс для спілкування з іншими продуктами та послугами. За останні декілька років, його використання стрімко зросло і багато найпопулярніших веб-програм сьогодні були б неможливі без API.

Прикладний програмний інтерфейс – це набір визначених правил, що пояснюють, як комп'ютери або програми взаємодіють між собою. API розташовані між додатком та веб-сервером, виконуючи роль проміжного рівня, який обробляє передачу даних між системами.

Цей механізм працює наступним чином (рис. 4.1):

– клієнтська програма ініціює виклик API для отримання інформації, також відомої як запит;

– після отримання коректного запиту, API здійснює виклик зовнішньої програми або веб-сервера;

– сервер надсилає відповідь на API із інформацією, що запрошувалась;

– API передає дані початковій програмі, яка ініціювала запит.

За задумом, API гарантують безпеку, оскільки їхня позиція посередника полегшує абстрагування функціональних можливостей між двома системами (споживачем та інфраструктурою, що надає послуги). Виклики API зазвичай

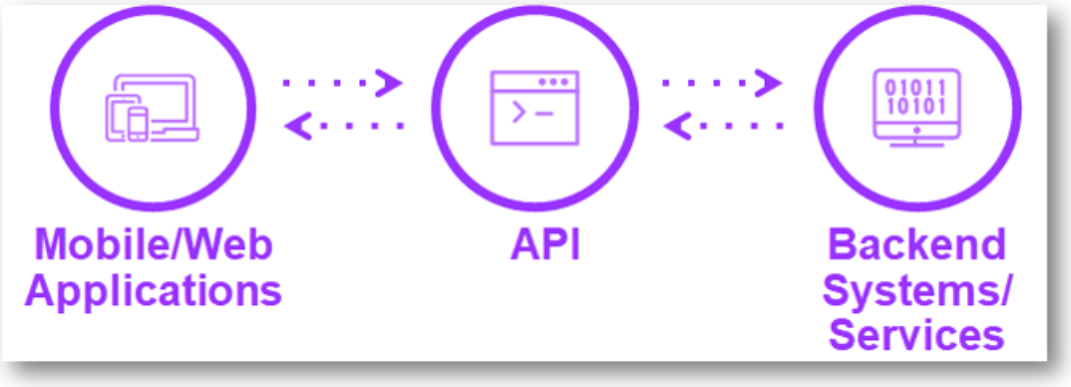


Рис. 4.1. Принцип роботи прикладного програмного інтерфейсу

містять облікові дані авторизації для зменшення ризику атак на сервер, а шлюз API може обмежувати доступ для мінімізації загроз безпеці. Також під час обміну заголовки HTTP, файли cookie або параметри рядка запиту забезпечують додаткові рівні захисту даних.

На сьогоднішній день більшість прикладних програмних інтерфейсів – це веб-API, які надають дані та функціональні можливості програми через інтернет. До основних типів веб-API можна віднести:

– відкриті API – інтерфейси з відкритим кодом, до яких можна отримати доступ за допомогою протоколу HTTP;

– партнерські API – інтерфейси, яким надаються стратегічним або діловим партнерам (зазвичай розробники можуть отримати доступ до цих API в режимі самообслуговування через загальнодоступний портал розробників API, але їм потрібно буде завершити процес підтвердження належності до компанії);

– внутрішні API – інтерфейси, які залишаються прихованими від зовнішніх користувачів (ці приватні API не доступні для користувачів за межами компанії, а натомість призначені для підвищення продуктивності та комунікації між різними внутрішніми командами розробників);

– складені API – інтерфейси, які поєднують в собі кілька API (вони корисні в архітектурі мікросервісів, де для виконання одного завдання може знадобитися інформація з кількох ресурсів).

Оскільки використання веб-API зросло, були розроблені певні протоколи, що забезпечують користувачам набір визначених правил, які визначають типи даних та команди. По суті, ці протоколи сприяють стандартизованому обміну інформацією.

SOAP (Simple Object Access Protocol) – це протокол прикладного програмного інтерфейсу, побудований на основі XML, що дозволяє користувачам надсилати та отримувати дані через SMTP та HTTP. З його допомогою легше обмінюватися інформацією між програмами або програмними компонентами, які працюють в різних середовищах або написані різними мовами.

XML-RPC – це протокол, який спирається на певний формат XML для передачі даних, тоді як SOAP використовує власний формат XML. XML-RPC старший за SOAP, але набагато простіший і відносно легкий, оскільки використовує мінімальну пропускну здатність.

JSON-RPC – це протокол, подібний до XML-RPC, але він використовує JSON замість формату XML для передачі даних. Хоча виклики можуть містити кілька параметрів, вони очікують лише одного результату.

REST (Representational State Transfer) – це набір принципів архітектури веб-API, що означає відсутність офіційних стандартів (на відміну від тих, що мають протокол). Щоб бути REST API (також відомий як RESTful API), інтерфейс повинен дотримуватися певних архітектурних обмежень. Можна побудувати RESTful API з протоколами SOAP, але ці два стандарти зазвичай розглядаються як конкуруючі специфікації.

При використанні API існують два загальноприйняті архітектурні підходи – сервісно-орієнтована архітектура (SOA), та архітектура мікросервісів.

Сервісно-орієнтована архітектура – це стиль побудови програмного забезпечення, у якому функції розділені та надані як окремі сервіси в мережі. Як правило, SOA реалізується з веб-службами, роблячи функціональні блоки доступними за допомогою стандартних протоколів комунікації.

Архітектура мікросервісів – це альтернативний архітектурний стиль, який розділяє додаток на менші, незалежні компоненти. Застосування програми як набору окремих служб полегшує тестування, підтримку та масштабування. Ця методологія стала популярною протягом усього часу хмарних обчислень, що дозволяє розробникам працювати над одним компонентом, незалежним від інших.

Хоча SOA був важливим еволюційним кроком у розробці додатків, архітектура мікросервісів побудована із урахуванням масштабування, забезпечуючи розробникам та підприємствам гнучкість, необхідну їм для створення, модифікації, тестування та розгортання програм на детальному рівні, з меншими циклами ітерацій та ефективним використання ресурсів хмарних обчислень [10].

Оскільки API дозволяють компаніям відкривати доступ до своїх ресурсів, зберігаючи при цьому безпеку та контроль, вони стали цінним аспектом сучасного бізнесу. Розглянемо кілька популярних прикладів, з якими можна часто зіткнутися.

Універсальні логіни. Популярний приклад API – це функція, яка дозволяє людям входити на веб-сайти, використовуючи свої дані для входу в профілі Facebook, Twitter або Google. Ця зручна функція дозволяє будь-якому веб-сайту використовувати API однієї з найпопулярніших служб для швидкої автентифікації користувача, економлячи їм час та клопоти щодо налаштування нового профілю для кожної служби веб-сайту.

Бронювання подорожей. Сайти для бронювання подорожей об’єднують тисячі рейсів, демонструючи найдешевші варіанти для кожної дати та пункту призначення. Ця послуга стала можливою завдяки API, які надають користувачам додатків доступ до найсвіжішої інформації про доступність готелів та авіакомпаній. Завдяки автономному обміну даними та запитами, API різко скорочує час та зусилля, необхідні для перевірки наявних рейсів або розміщення.

Карти Google. Одним з найпоширеніших прикладів хорошого API є сервіс Google Maps. На додаток до основних API, які відображають статичні або інтерактивні карти, програма використовує інші API та функції, щоб надавати користувачам вказівки або цікаві місця. За допомогою геолокації та декількох шарів даних ви можете спілкуватися з API Maps, коли складаєте маршрути.

Twitter. Кожен твіт містить описові основні атрибути, включаючи автора, унікальний ідентифікатор, повідомлення, позначку часу, коли він був опублікований, і метадані геолокації. Twitter робить загальнодоступні твіти та відповіді доступними для розробників і дозволяє розробникам розміщувати твіти за допомогою API компанії.

4.2 Прикладний програмний інтерфейс веб-сервісу із розкладом занять

Розроблене програмне забезпечення використовує у роботі прикладний програмний інтерфейс веб-сервісу із розкладом mytimetable.live Для передачі даних у ньому застосовується формат JSON. Поля за замовчуванням, що приходять у відповідь на запит позначені у таблиці 4.1, стандартні параметри запиту – в таблиці 4.2 [11].

Дане API дає змогу отримати всю необхідну інформацію для формування розкладів та звітів, а саме інформацію про університети, факультети, спеціальності, курси, групи, корпуси, аудиторії, вчителів, семестри, навчальні роки, розлад групи, розклад початку і кінця занять, предмети, іспити, заліки тощо.

Для отримання необхідної інформації існує відповідне посилання із параметрами, якими можна фільтрувати результати запиту. Наприклад для отримання списку усіх університетів, які доступні у базі даних, необхідно зробити GET HTTP-запит за адресою «https://api.timetable.pp.ua/rest/universities».

Розглянемо детальніше роботу прикладного програмного інтерфейсу веб-ресурсу для кожного запиту, необхідного при формуванні звітів.

Таблиця 4.1

Назва поля	**Пояснення**	**Приклад**
count	Кількість записів у відповідь. Позитивне ціле число, максимальне значення 100.	10
next	Посилання на наступну сторінку із результатами запиту	Поточна сторінка: /rest/universities/?limit=1&offset=1 Наступна сторінка: /rest/universities/?limit=1&offset=2
previous	Посилання на попередню сторінку із результатами запиту	Поточна сторінка: /rest/universities/?limit=1&offset=1 Попередня сторінка: /rest/universities/?limit=1
results	Список із результуючими записами	[{ "url": "http://localhost:8000/rest/universities/1/", "id": 1, "__unicode__": "Київський Національний Університет", "name": "Київський Національний Університет", "short_name": "КНУ", "img": null, "desc": "Description" }]

Таблиця 4.2

Назва параметру	Призначення	Синтаксис	Приклад
fields	Визначає список полів, які будуть повернуті у відповідь.	Список полів, розділених комою. Назви полів повинні дорівнювати полям, які містить відповідь.	?fields=url,id
limit	Обмежує результат запиту (дорівнює обмеженню SQL).	Позитивне ціле число. Максимальне значення 100.	*?limit=5*
offset	Зсув результатів запиту.	Позитивне ціле число.	*?offset=5*

Університети. Як отримати список із університетів було описано вище. В полі «results» знаходяться усі результуючі записи із наступною інформацією по кожному навчальному закладу: url, id, __unicode__, повна назва закладу, коротка її назва, посилання на зображення, опис та коротка назва англійською. Для фільтрації результатів і уточнення запиту, можна використати параметр ?search, вказавши повність або частково назву університету. Наприклад, https://api.timetable.pp.ua/rest/universities/?search=шев. Цим запитом можна отримати усі навчальні заклади у яких в назві фігурує «шев».

Факультети. До API треба звернутись за посиланням https://api.timetable.pp.ua/rest/faculties/. Так само, як і для університетів, у JSON відповіді в полі «results» можна знайти факультети, які запитувались. Поля з інформацією аналогічні попередньому пункту, тільки адаптовані від факультети, до яких додається ще одне – id університету, до якого відноситься цей факультет. Окрім пошуку в назвах, є фільтрація за університетом. Для цього треба

використати параметр ?univ та вказати id університету. Наприклад ?univ=7, щоб отримати всі факультети КНУТД.

Спеціальності. Посилання: https://api.timetable.pp.ua/rest/specialties/. Інформаційні поля для спеціальностей схожі до факультетів, з однією відмінністю, а саме замість id університету використовується поле з id факультету. Фільтрація результатів можлива із використанням параметрів ?univ та ?faculty, що відповідають за id університету та факультету відповідно. Приклад запиту із використанням цих параметрів: https://api.timetable.pp.ua/rest/specialties/?faculty=7&univ=8. Цим запитом відфільтровуються спеціальності, які належать до факультету «Інформатики» Національного університету «Києво-Могилянська академія».

Курси. Посилання: https://api.timetable.pp.ua/rest/courses/. До полів JSON результату відносяться: url, id, __unicode__, назва курсу, освітній кваліфікаційний рівень (від 0 до 3, що позначають «бакалавр», «магістр», «спеціаліст» та «інші» відповідно), id спеціальності, якій відповідає даний курс. Параметри, які можна застосувати для пошуку: ?univ, ?faculty, ?specialty, ?degree (id університету, факультету, спеціальності та ОКР). Знайти усі курси КНУ, факультету КНК та спеціальності «Штучний інтелект», що навчаються на магістратурі можна наступним запитом до API: https://api.timetable.pp.ua/rest/courses/?degree=1&specialty=18&faculty=1&univ=1.

Групи. Посилання: https://api.timetable.pp.ua/rest/groups/. Поля, що відрізняються від попередніх: назва, коротка назва, підгрупа, коротка назва англійською, назва курсу, ОКР, id навчального року та id курсу. Фільтрувати запит можна за допомогою навчального року, курсу, спеціальності, факультету, університету та ОКР (?year, ?course, ?specialty, ?faculty, ?univ, ?degree відповідно).

Корпуси. Посилання: https://api.timetable.pp.ua/rest/housings/. Доступна інформація про кожен корпус: назва, коротка назва, адреса, локація, кількість поверхів та id університету. Параметри для керування результатами запиту: університет та факультет (?univ, ?faculty).

Аудиторії. Посилання: https://api.timetable.pp.ua/rest/rooms/. Для кожної аудиторії API повертає url, id, __unicode__, назву, поверх та id. Із доступних фільтрів наявні такі параметри: університет, факультет та корпус (?univ, ?faculty, ?housing).

Викладачі. Посилання: https://api.timetable.pp.ua/rest/teachers/. Поле «results» відповіді на такий тип запитів повертає повне ПІБ викладача, його скорочену форму, науковий ступінь (0 – викладач, 1 – старший викладач, 2 – доцент, 3 – професор, 4 – асистент, 5 - інші), посилання на зображення, опис, коротка назва англійською та id університету. Запити параметризуються за допомогою наукового ступеню, університету та групи (?degree, ?univ, ?group).

Навчальний рік. Посилання: https://api.timetable.pp.ua/rest/years/. Відповідь на запит характеризується початком і кінцем навчального року та id університету. Фільтрується запит тільки за допомогою поля університету (?univ).

Семестри. Посилання: https://api.timetable.pp.ua/rest/semesters/. Кожен семестр повертається із полями: номер семестру, id навчального року, його початок та кінець. Фільтруються результати за допомогою навчального року та групи (?year, ?group).

Предмет. Посилання: https://api.timetable.pp.ua/rest/lessons/. Для кожного запиту повертається список з предметами, які описані полями: id корпусу, id аудиторії, id часу проведення заняття, формат (0 – лекція, 1 – семінар, 2 – практика, 4 – лабораторне заняття, 5 - інше), день (0-5, що відповідають понеділку-суботі відповідно), підгрупа, тижні проведення, список із id викладачів, назва, тип проведення (дистанційно, очно). Керувати результатами запитів можна використавши параметри: назва, корпус, аудиторія, формат, день, група/семестр (?theme, ?housing, ?room, ?format, ?day, ?group_semester).

Час проведення заняття. Посилання для цього типу запитів: https://api.timetable.pp.ua/rest/lessontime/. Список результату містить для кожного запису час початку і кінця проведення, наявність перерви, початок і кінець перерви та id факультету. Відфільтрувати результати можна за допомогою факультету (?faculty).

Розклад групи. Для цього типу запитів необхідно заздалегідь знати коротку назву англійською, щоб сформувати посилання для запиту до API. Наприклад для отримання розкладу групи ІПЗ-1 Києво-Могилянської академії, коротка англійська назва якої «IPZ-1-naukma», IPZ-1-naukma звернутися за наступною адресою: https://api.timetable.pp.ua/rest/timetable/?group=IPZ-1-naukma. У JSON полі «lessons» знаходиться список із предметів, що викладаються цій групі. Для кожного елементу цього списку повертаються: id, повна назва, коротка назва, корпус (id, назва, коротка назва, локація), аудиторія (id, номер, поверх), тип проведення, час проведення заняття, формат, підгрупа, вчителі (список із викладачами, кожен із яких з полями: id, повне ПІБ, коротке ПІБ, посилання на зображення, науковий ступінь, назва англійською), список із датами проведення, тижні проведення, факультет (id, назва, коротка назва, зображення, назва англійською). У полі «lesson_time» повертається список із часом проведення занять, де для кожного запису є id, час початку, кінця, наявність перерви та час початку і кінця перерви. Поле «info» відповіді містить групу (id, назва, коротка назва, назва англійською, підгрупа), університет (id, назва, коротка назва, посилання на зображення, коротка назва англійською), факультет (id, назва, коротка назва, посилання на зображення, коротка назва англійською), спеціальність (аналогічні поля до факультету), курс (id, назва, ОКР), семестр (id, номер, початок і кінець), навчальний рік (id, початок і кінець). Фільтрація результатів для цього типу запитів не передбачена.

Розклад сесії групи. Аналогічно до розкладу групи, потрібно наперед знати коротку назву англійською. Приклад запиту для групи ПЗС-2 факультету КНК: https://api.timetable.pp.ua/rest/controls-custom/?group=PZS-2. Поле «info» аналогічне попередньому типу запитів. Поле «controls» складається зі списку усіх предметів, для яких проводиться контроль. Кожен предмет складається із id, повної назви предмета, короткої назви предмета, корпусу (id, назва, коротка назва, локація), аудиторія (id, номер, поверх), формату проведення (1 – залік, 2 – консультація, 3 – залік, 4 – захист, 5 – комісія, 6 – практика, 7 – курсова, 8 – перескладання, 0 - інші), підгрупи, списку викладачів (такі ж поля, як і в запиті

розкладу групи), дати проведення, часу початку, типу проведення, факультету (id, назва, коротка назва, посилання на зображення, коротка назва англійською). Відфільтрувати результати також неможливо.

РОЗДІЛ 5
РОЗГОРТАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ НА СЕРВЕРІ

5.1 Інтерфейс користувача

Користувацький інтерфейс складається із однієї веб-форми, яка динамічно змінюється відповідно до обраних параметрів. На ній присутні дві секції: основні та додаткові параметри (рис. 5.1).

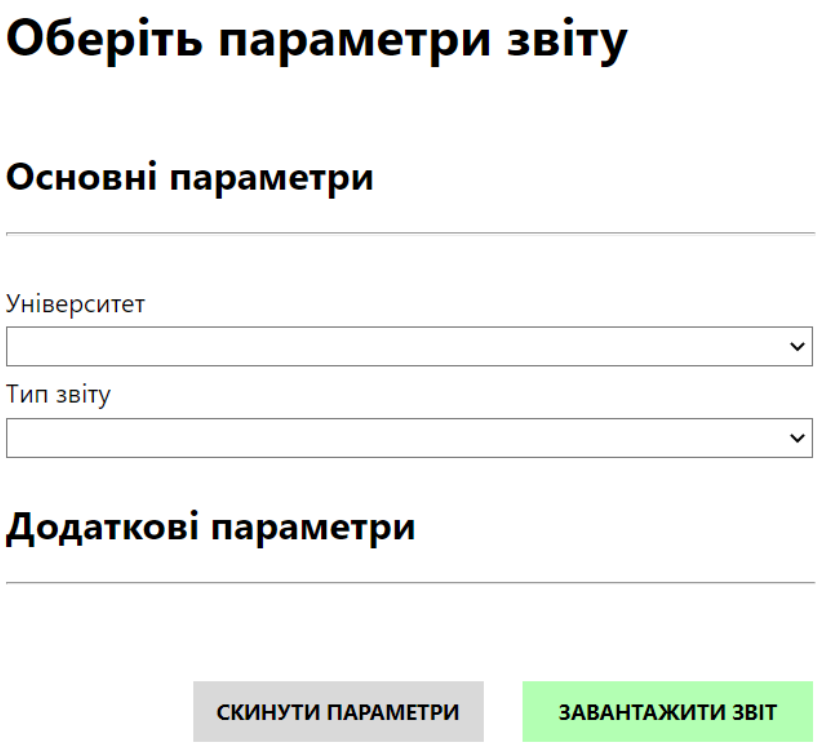


Рис. 5.1. Інтерфейс користувача

Секція основних параметрів складається із двох випадаючих списків: «Університет» та «Тип звіту». Університети, для яких доступні розклади:

– Національний університет «Києво-Могилянська академія»;
– Київський національний університет технологій та дизайну;
– Київський національний університет імені Тараса Шевченка;

– Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського».

У секції «Додаткові параметри», відповідно до обраного типу звіту, відображаються наступні випадаючі списки:

а) «Розклад групи»;

– «Факультет»;
– «Спеціальність»;
– «Група»;

б) «Розклад факультету»;

– «Факультет»;
– «ОКР»;

в) «Розклад аудиторій факультету»;

– «Факультет»;

г) «Розклад сесії для групи»;

– «Факультет»;
– «Спеціальність»;
– «Група»;

ґ) «Розклад сесії для курсу» та «Розклад курсу»;

– «Факультет»;
– «ОКР»;
– «Номер курсу»;

д) «Розклад аудиторій корпусу»;

– «Корпус»;

е) «Розклад спеціальності»;

– «Факультет»;
– «Спеціальність»;

є) «Розклад викладачів, що читають пари на спеціальності»;

– «Факультет»;
– «Спеціальність».

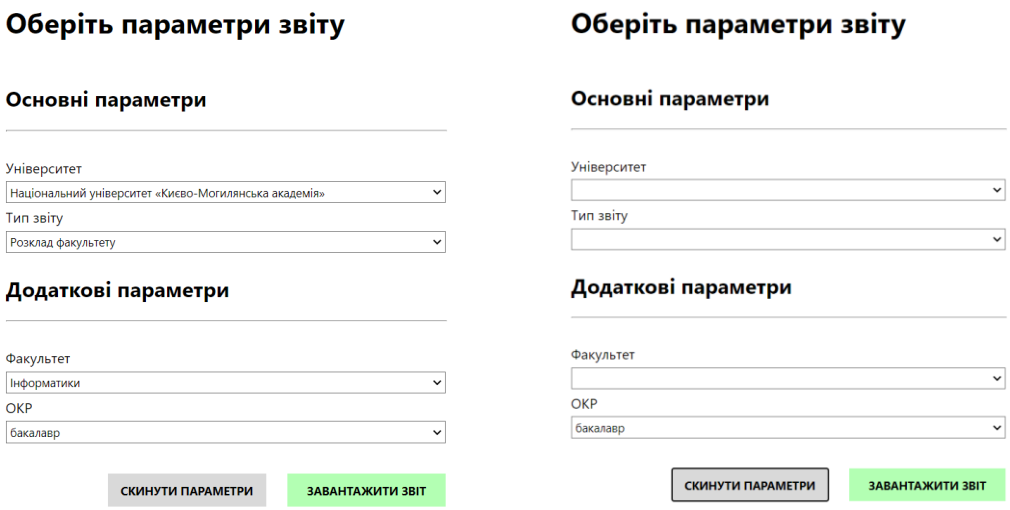
Для звітів «Розклад викладача», «Розклад сесії викладача», «Розклад викладачів, що читають пари на спеціальності» у додаткових параметрах відображається поле вводу «Викладач», у якому по введеним першим літерам фільтруються прізвища викладачів, наявних у базі.

Для звітності розкладу аудиторій університету додаткових параметрів немає, тож жодного поля у цій секції не відображається.

На формі присутні дві кнопки: «Скинути параметри» та «Завантажити звіт». За допомогою першої, користувач може скинути усі обрані параметри на формі та розпочати все спочатку (рис. 5.2). Друга кнопка відправляє на обробку введені дані та надає очікуваний звіт.

Форма також перевіряє правильність введення даних при спробі завантажити звіт. Для кожного типу перевіряється обов'язкове поле, яке повинно бути вибраним. Такими полями є:

– «Група», для розкладу групи та її сесії;
– «Факультет», для розкладу факультету та аудиторій;
– «Номер курсу», для розкладу сесії курсу;
– «Викладач», для розкладу викладача;
– «Спеціальність», для розкладу груп спеціальності та їх викладачів.



а) б)

Рис. 5.2. Робота кнопки «Скинути параметри»: а – до натискання; б) – після натискання

Якщо ж при спробі завантажити результуючий файл, тип звіту не був обраний, то система видає відповідне повідомлення, щодо його необхідності. Дана форма була інтегрована у панель адміністратора ресурсу api.mytimetable.live, тож до неї можна потрапити натиснувши на посилання позначеному на рис. 5.3 червоним прямокутником.

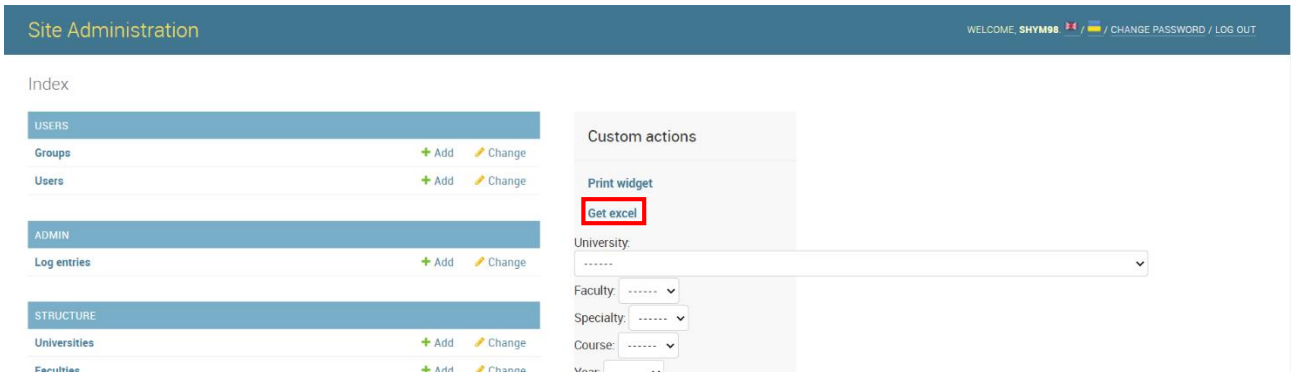


Рис. 5.3. Панель адміністратора

5.2 Процес розгортання програмного засобу

При розробці використовувалась безперервна інтеграція. Це стратегія розробки програмного забезпечення, що збільшує швидкість розробки, при цьому забезпечуючи якість коду, що розгортається. Постійно робляться коміти (принаймні кожного дня або й кілька разів на день), які потім автоматично збираються та перевіряються перед тим, як об’єднати із репозиторієм.

Тому для цього була необхідність в застосуванні додаткових інструментів розробки. Таким став Drone CI. Найвизначнішою особливістю Drone є те, що він використовує контейнери для всього. Кожен етап у Drone виконується через контейнер Docker. Це пропонує велику гнучкість, коли мова заходить про використання декількох інструментів та/або середовищ для збірки та розгортання. На відміну від Jenkins, Drone CI повинен інтегруватися із репозиторієм Git [12], щоб нормально функціонувати. Він має ряд плагінів, які можна розгорнути як контейнери Docker [13].

Drone CI використовує файли YAML для отримання своїх інструкцій. Файл інструкцій перевіряється у репозиторії разом із рештою коду програми. Ця поведінка дуже схожа на декларативні конвеєри в Jenkins, коли вони налаштовані на використання його файлу, який є частиною репозиторію.

Для хостингу проекту використовувався веб-сервіс Bitbucket. Він оснований на системі контролю версій Git [14]. Одним із плюсів вибору Bitbucket є те, що Drone CI підтримує Bitbucket Pipelines [15], що полегшує процес неперервної інтеграції.

Тож процес розгортання можна описати наступним чином:

– відбувається команда «push» у гілку dev або master репозиторію;
– спрацьовує тригер і CI виконує збірку нового контейнера;
– контейнер проходить тестування;
– при успішному проходженні перевірки, контейнер відправляється на віртуальну машину, що розташована на Digital Ocean;
– там старий контейнер видаляється та розгортається новий.

ВИСНОВКИ

В результаті виконання кваліфікаційної роботи було визначено типи звітів про розклад та розроблено їх шаблони, досліджено особливості взаємодії з прикладним програмним інтерфейсом веб-сервісу, реалізовано програмний продукт та розгорнуто на сервері.

Даний програмний засіб може застосовуватися навчальними закладами, яким необхідно створювати різного виду звіти про розклад у вигляді електронних таблиць. Оскільки його функціональність можна легко розширити для довільної кількості нових шаблонів, то є доцільним продовження дослідження та розробки в цій області.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. David M. Beazley. Python Essential Reference. — 4th Edition. — Addison-Wesley Professional, 2009. — pp: 60-63.
2. Creating Excel files with Python and XlsxWriter. [Електронний ресурс]. – Режим доступу: https://xlsxwriter.readthedocs.io/
3. Python Resources for working with Excel. [Електронний ресурс]. – Режим доступу: https://www.python-excel.org/
4. Requests: HTTP for Humans™. [Електронний ресурс]. – Режим доступу: https://docs.python-requests.org/en/master/
5. HTTP Request Methods – What are HTTP Requests?. [Електронний ресурс]. – Режим доступу: https://rapidapi.com/blog/api-glossary/http-request-methods/
6. Django: The Web framework for perfectionists with deadlines. [Електронний ресурс]. – Режим доступу: https://www.djangoproject.com/
7. Django Project MVT Structure. [Електронний ресурс]. – Режим доступу: https://www.geeksforgeeks.org/django-project-mvt-structure/
8. Introduction to Django Views. [Електронний ресурс]. – Режим доступу: https://www.pluralsight.com/guides/introduction-to-django-views
9. Application Programming Interface (API). [Електронний ресурс]. – Режим доступу: https://www.ibm.com/cloud/learn/api
10. Microservices vs SOA: What's the Difference?. [Електронний ресурс]. – Режим доступу: https://dzone.com/articles/microservices-vs-soa-whats-the-difference
11. MyTimeTable REST/ Api Root Documentation [Електронний ресурс]. – Режим доступу: https://api.timetable.pp.ua/rest/documentation/
12. Building A CD Pipeline With Drone CI And Kubernetes. [Електронний ресурс]. – Режим доступу: https://www.magalix.com/blog/building-a-cd-pipeline-with-drone-ci-and-kubernetes
13. Building Docker Images using Drone. [Електронний ресурс]. – Режим доступу: https://www.katacoda.com/courses/cicd/build-docker-images-drone

14. Bitbucket | The Git solution for professional teams. [Електронний ресурс]. – Режим доступу: https://bitbucket.org/

15. Bitbucket Pipelines & Deployments. [Електронний ресурс]. – Режим доступу: https://bitbucket.org/product/features/pipelines