Курсовий проект

Виконав: Ларіонов Олександр Олександрович

Група: КМ-83

Тема: Моделювання освітньої платформи

Зміст

Вступ

В роботі змодельовано універсальну освітню платформу й основні її компоненти. Суть в наступному:

На платформі можуть реєструватися різні освітні компанії (школи / секції / держ установи) (з потенціалом подальшої монетизації). Адміністратори кожної окремої компанії зможуть налаштувати внутрішню ізольовану інфраструктуру під свої потреби. Це такі речі, як курси, групи, навчальні дисципліни, заплановані уроки, студенти, викладачі і тд.

На базі цього буде налагоджена робота компанії з клієнтами по принципу «все в одному місці» (уроки, домашні завдання, історія оплат, викладачі). Окрім того, компанія отримає автоматичний розрахунок зарплат викладачам і оплат клієнтів (потрібно буде лише задати певні параметри), а також в потенціалі автоматична генерація звітів / графіків і подібного.

Компанії зможуть відслідковувати найефективніших викладачів і найприбутковіших клієнтів, і робити відповідні зміни/пропозиції.

В цій роботі деякий функціонал було зроблено у вигляді прототипу, а більше роботи було виділено на такі речі, як прорахування оплат клієнтів, зарплат викладачам, і генерації діаграм результату компаній та окремих груп всередині.

Платформа є сайтом, тож користувачі повинні мати лише браузер, щоб почати нею користуватися.

Постановка задачі

Цільові задачі наступні:

- Розробити схему бази данних
- Продумати й реалізувати ізоляцію даних всередині компаній (як в базі даних, так і з точки зору функціоналу)
- Можливість входити в компанію
- Проробити головну сторінку компанії
- Генерувати діаграми для аналізу роботи компанії
- Змоделювати автоматичне підлаштування зарплатні та оплати клієнтів в залежності від кількості студентів у групі
- Деталізувати створення й перегляд предметів усередині компанії
- Сторінка груп де потрібно виводити детальну інформацію по кожній групі та її учасників
- Другорядні CRUD – операції - розробити у вигляді адмін-панелі, де адміністратор може редагувати будь-які дані
- Вираховувати чисту виручку компанії за кожен місяць (в тому числі за кожну групу)

Основними вимогами при розробці є:

- Користувач може користуватися платформою з браузера
- Реалізуваи паралельну роботу користувачів на платформі
- Реалізувати розділення користувачів по ролям в кожній компанії

Проектування математичного забезпечення

Основною задачею являється моделювання доходів та витрат компаній таким чином, щоб компанія отримувала достатній прибуток. Задачу ускладнює те, що при моделюванні варто врахувати різну кількість студентів в кожній групі, а також різну відвідуваність студентами занять.

Це буде мати вплив як на запрлатню викладачів, так і на те, скільки матимуть платити студенти (студентами можуть бути як дорослі люди, так і діти).

Чим більше студентів займається у групі, тим більшу зарплатню має отримувати викладач, і одночасно менше має платити кожен студент. Більш того, зі зростом кількості студентів компанія має пропорційно більше заробляти.

Тобто нам треба врахувати граничну вартість кожного студента на групу і розрахувати все так, щоб усі отримали вигоду.

Гранична вартість – один з основних методів метрологічного (кількісного) відображення величини вартості (цінності, ціни) товарів, послуг, інших інгредієнтів матеріального і духовного виробництва. Гранична вартість показує, чого варта кожна додаткова одиниця певного товару.

$$CM(Q) = dC / dQ$$

Відповідно, спробуємо змоделювати оптимальну формулу підрахунку зарплат і витрат клієнтів.

Була змодельована наступна система:

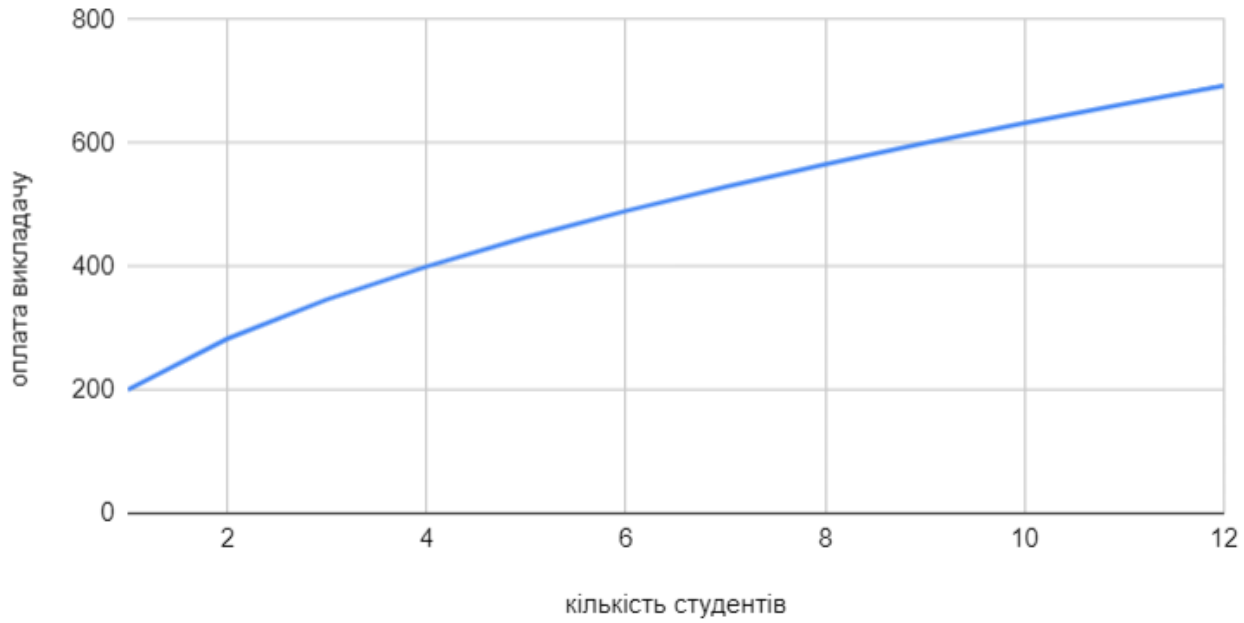
Для кожного курсу в компанії задається початкова ціна для викладача та початкова ціна для студента. Далі ціна для деякої кількості студентів N рахується за формулою, яка залежить від кількості студентів у групі та початкової ціни P. Спробуємо підібрати відповідну формулу.

Спочатку проаналізуємо оплату викладачу

Стартова ціна	кількість студентів	оплата викладачу
200	1	200
200	2	282,8427
200	3	346,4102
200	4	400
200	5	447,2136
200	6	489,8979
200	7	529,1503
200	8	565,6854
200	9	600
200	10	632,4555
200	11	663,325
200	12	692,8203

Була підібрана наступна формула:

$$T = P * \sqrt{N}$$



Це витягнута $y = \sqrt{x}$

Відповідно, гранична корисність кожного нового студента в проекції на зарплатню викладачу меншає, але його зарплатня росте достатньо, щоб той мотивувався вести у більшої кількості студентів.

Тепер проаналізуємо оплату студента за заняття:

Стартова ціна	кількість студентів	оплата від студента	сумарна оплата
350	1	350	350
350	2	247,4873734	494,9747
350	3	202,0725942	606,2178
350	4	175	700
350	5	156,5247584	782,6238
350	6	142,8869017	857,3214
350	7	132,2875656	926,013
350	8	123,7436867	989,9495
350	9	116,6666667	1050
350	10	110,6797181	1106,797
350	11	105,5289706	1160,819
350	12	101,0362971	1212,436

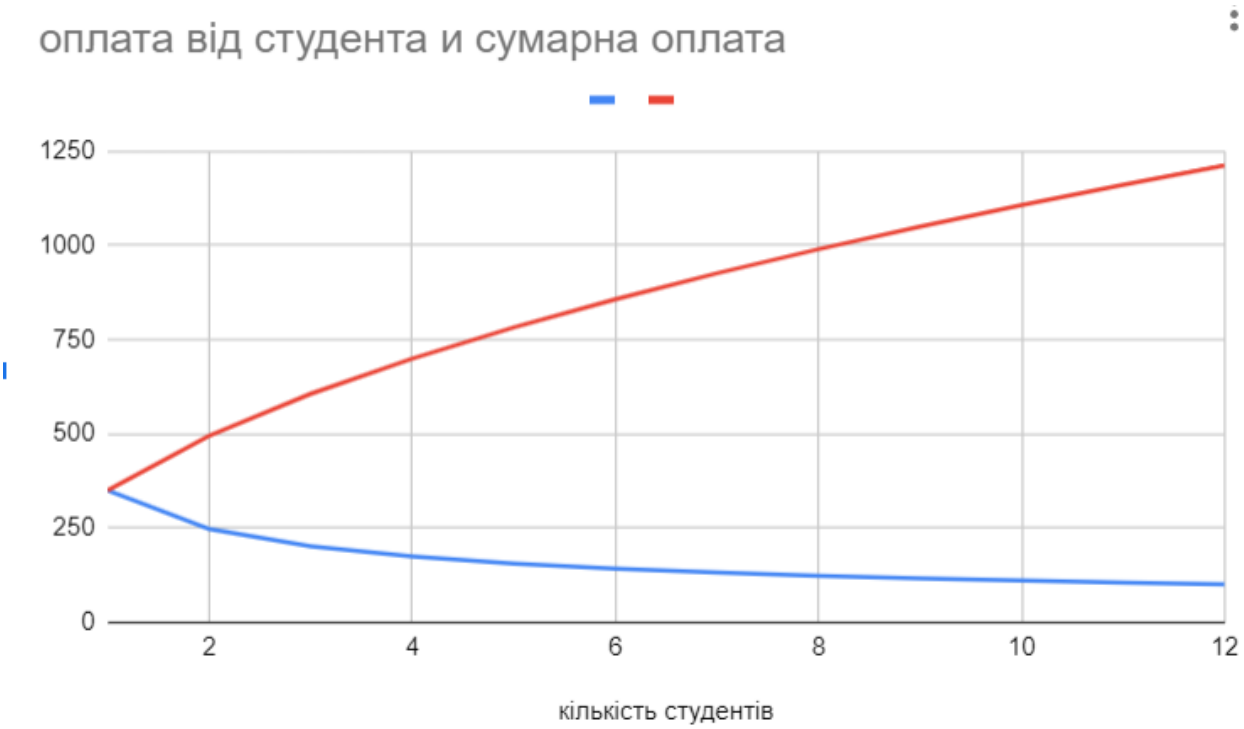
Була підібрана наступна формула:

$$T = \frac{P}{\sqrt{N}}$$

T – скільки має платити клієнт

P – початкова ціна

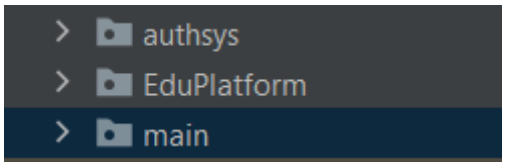
N – кількість студентів у групі



Тут червона лінія – виручка компанії за урок з відповідною кількістю студентів. Синя – витрати студентів. Як можна побачити, студенти, як і компанія, в плюсі від того, чим більше студентів в групі.

Архітектура розроблених програмних засобів

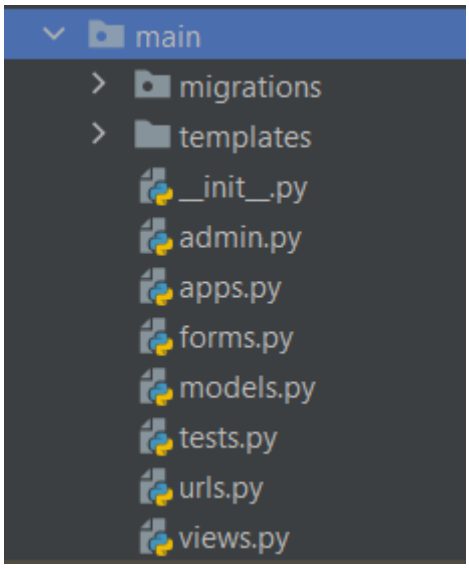
Сайт розроблювався мовою Python з використанням фреймфорка Django. Відповідно, структура проекту доволі велика.



В корені проекта маємо 3 папки – це 3 застосунки Django.

- EduPlatform –застосунок, що відповідає за налаштування проекту
- Authsys – модуль користувачів. Тут задані таблиці для бази даних користувачів, а також форми реєстрації та авторизації.
- Main – головний застосунок, в якому реалізована логіка роботи з компаніями

Зануримося в головний застосунок - мейн



Тут бачимо набір файликів:

- Models.py - тут зберігаються моделі проекту. Це відображення таблиць у базі даних. Це – обгортка, що дозволяє керувати рядками в базі даних, як ООП-шними об’єктами.
- Views.py – файл з контроллерами сайту. Тут лежить основна логіка відповіді на запити користувачів
- Urls.py – файл з маршрутизацією. Тут прописані всі url- маршрути до потрібних контролерів застосунку.
- Forms.py – файл з формами
- Admin.py – файл для налаштувань адмін панелі.

- Migrations – папка з міграціями бази даних застосунку. Завжди можна відкатитися до попередньої версії схеми бази даних.
- Templates – папка з html – шаблонами застосунку

Основна логіка:

Запит від користувача направляється на потрібний маршрут в файлі маршрутів.

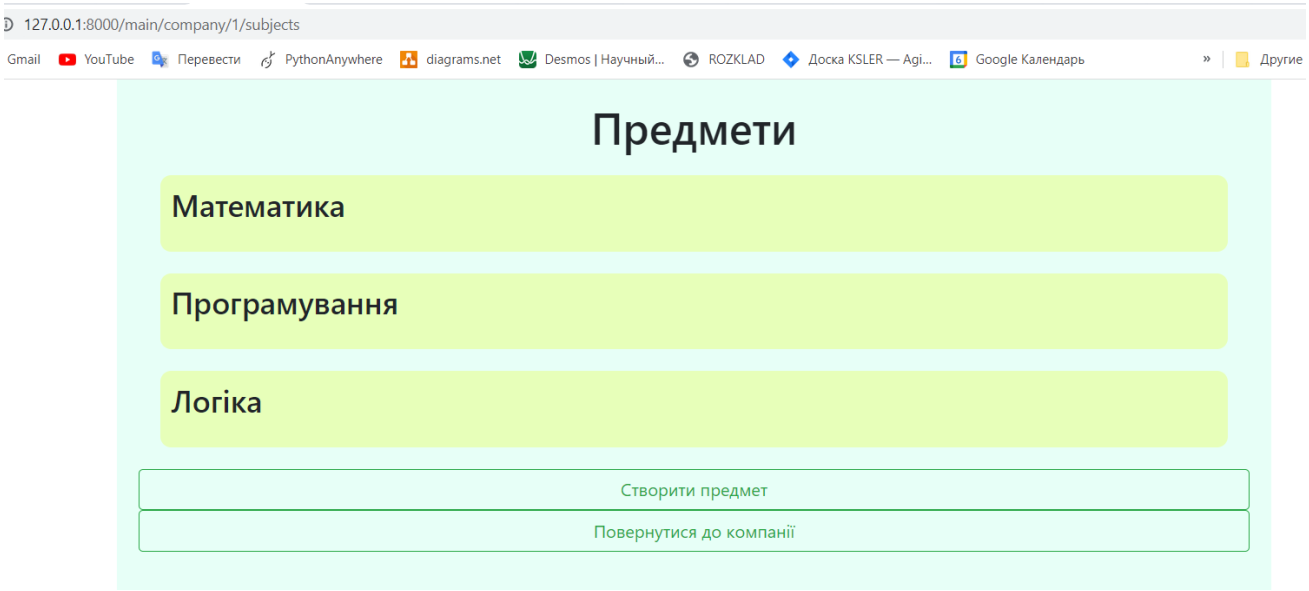
```python
from .views import *


app_name = 'main'

urlpatterns = [
    path('my_companies', my_companies, name='my_companies'),
    path('company_admin/<int:comp_user_id>', lambda: None),
    path('company_teacher/<int:comp_user_id>', lambda: None),
    path('company_student/<int:comp_user_id>', lambda: None),
    path('enter_company/<int:id>', enter_company, name='enter_company'),
    path('enter_company/<int:id>', enter_company, name='enter_company'),
    path('company/<int:id>/subjects', subjects, name='subjects'),
```

Далі він направляється на відповідний контролер, наприклад, контролер для редагування дисциплін в компанії:

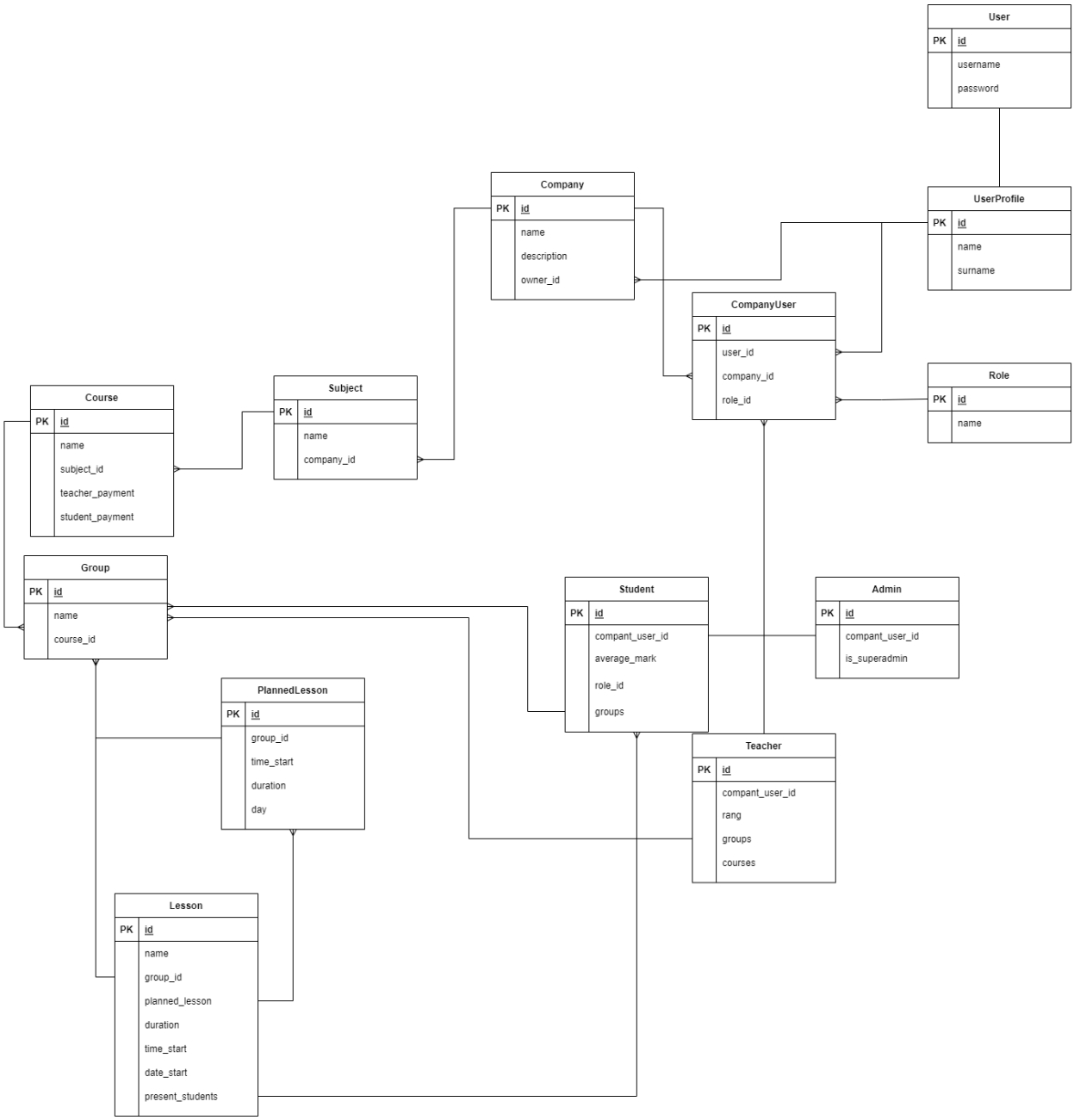
```python
def edit_subject(request, id, subject_id):
    subject = Subject.objects.get(id=subject_id)
    comp_user = CompanyUser.objects.get(id=id)
    if request.method == 'POST':
        form = SubjectForm(request.POST, instance=subject)
        if form.is_valid():
            form.save(comp_user.company_id)
            return redirect('main:subjects', id=id)
        else:
            print('invalid form')
    else:
        form = SubjectForm(instance=subject)

        return render(request, 'edit_subject.html', {'form': form, 'comp_user': comp_user, 'subject': subject}
```

Після цього генерується html-файл з шаблону, наприклад, subject_list.html.

І відповідно, в браузері ми бачимо сторінку


127.0.0.1:8000/main/company/1/subjects
Gmail
YouTube
Перевести
PythonAnywhere
diagrams.net
Desmos | Научный...
ROZKLAD
Доска KSLER — Agi...
Google Календарь
Другие
Предмети
Математика
Програмування
Логіка
Створити предмет
Повернутися до компанії

Також тут варто додати спроектовану схему бази даних:



Тут доволі багато таблиць, пов’язаних з користувачами, але це необхідно для реалізації можливості одному користувачу, наприклад, бути викладачем в одній компанії в другій бути студентом, а в третій бути одночасно і адміністратором, і викладачем.

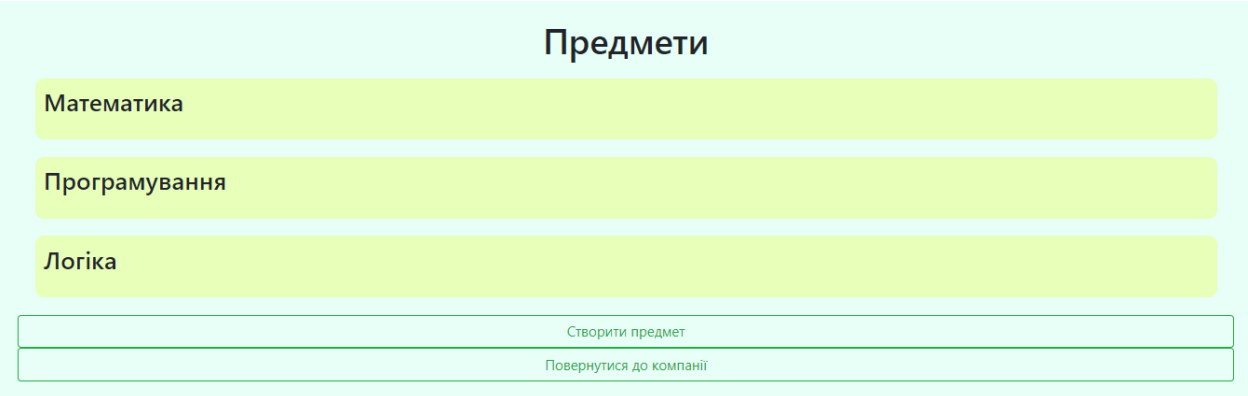
(! Це варто враховувати, якщо буде бажання створювати нових студентів в групах для тестування)

Опис інтерфейсу користувача

Інтерфейс користувача:

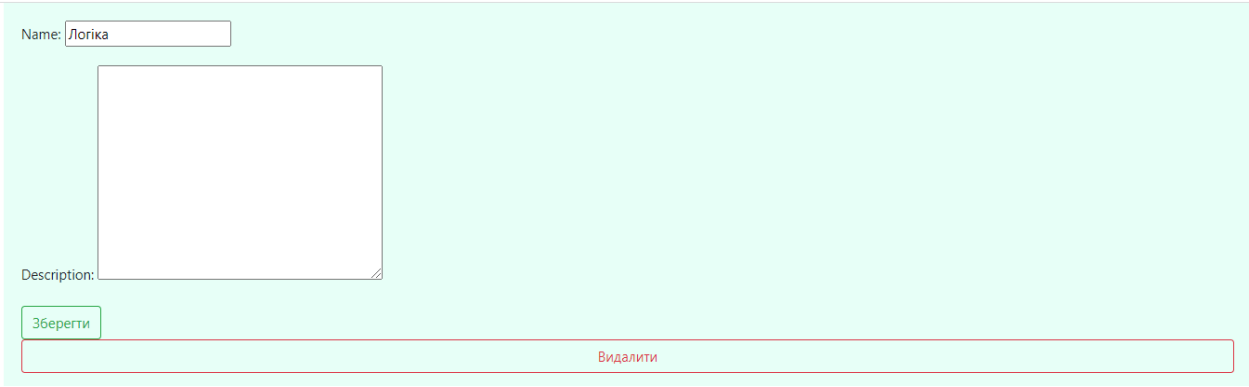
- Сторінка компанії

- 5 кнопок на цій сторінці
- Сторінка зі списком дисциплін та кнопками для додавання нових, і можливості редагування вже існуючих.



(тут послідовність така: Компанія->Дисципліна -> Курс -> Група)

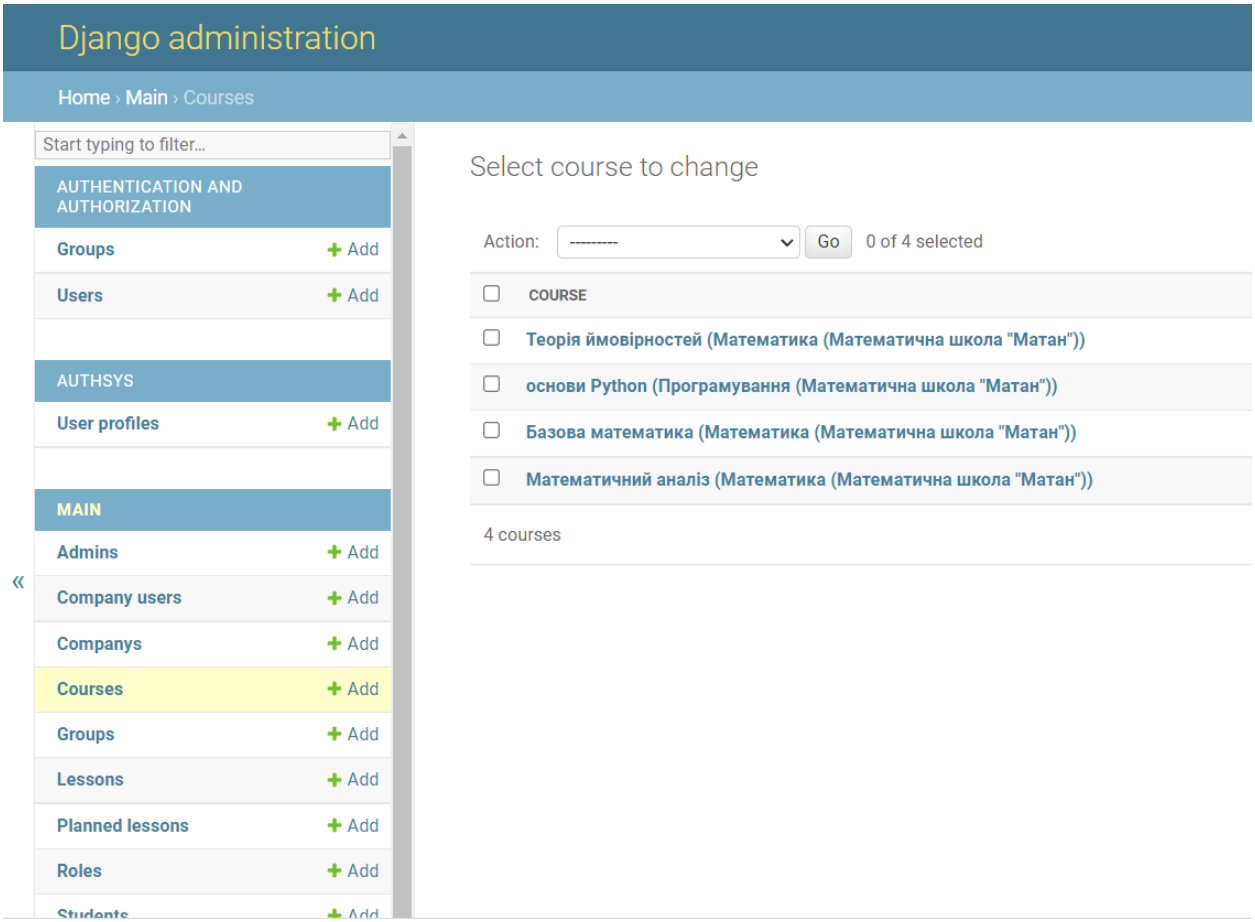
- Проста сторінка редагування предмету



- Сторінка з аналізом груп компанії

(на картинці сильно зменшив, що було зрозуміло загальна структура сторінки: на ній відображені всі групи з детальними даними й діаграмами по кожній)

- Адмін-панель, де користувач може редагувати будь-які дані, в тому числі, створювати уроки (з яких буде нараховуватися витрати й доходи)



Програмна реалізація

Один із простих але важливих алгоритмів – алгоритм розпізнавання ролі користувача при входженні в компанію

```
def enter_company(request, id):
    comp_users =
request.user.userprofile.companyuser set.filter(company id=id)
    comp_user = comp_users[0]
    role_name = comp_user.role.name
    return {
        'admin': company_admin,
        'teacher': company_teacher,
        'student': company_student,
    }[role_name](request, comp_user.id)
```

Тут дістаються дані користувача, разом із його основною роллю в цій компанії. Далі відбувається виклик функції, відповідної до ролі. Зараз основна роль – адмін.

```
def company_admin(request, comp_user_id):
    comp_user = CompanyUser.objects.get(id=comp_user_id,
userprofile__user__id=request.user.id)
    company = comp_user.company

    date = get_1day_date(datetime.date.today())
    months = {}
    for _ in range(6):
        income = company.calculate_income_for_month(date)
        spends = company.calculate_spends_for_month(date)
        value = income - spends
        months[f"{date.year}-{date.month}"] = {'income': income, 'spends':
spends, 'value': value}
        date = get_1day_date(date - datetime.timedelta(days=25))

    translate = {
        'income': 'Прибуток',
        'spends': 'Витрати',
        'value': 'Чиста виручка'
    }

    diags = []
    for label, ua_label in translate.items():
        diags.append(go.Bar(x=[month for month in months], y=[data[label] for
data in months.values()], name=ua_label))
    fig = go.Figure(data=diags)
    graph_html = fig.to_html()

    return render(request, 'company_admin.html', {'company': company,
'comp_user': comp_user, 'graph_html': graph_html})
```

Це – алгоритм збору інформації про оплати по компанії й відповідна побудова графіка.

Збираються дані з усіх груп компаніях за останні 6 місяців, після чого погружаються в відповідну структуру даних months, і за допомогою модуля plotly генерується html- код діаграми.

Після чого лишається влаштувати цей код в основний html-шаблон company_admin.html.

Схожий алгоритм можна побачити в контроллері, що відповідає за виведення інформації по групам.

```
def groups(request, id):
    comp_user = CompanyUser.objects.get(id=id)
    company = comp_user.company
    groups = company.groups

    result = []
    for group in groups:
        date = get_1day_date(datetime.date.today())
        months = {}
        for _ in range(6):
            income = group.calculate_students_payment(date)
            spends = group.calculate_teacher_payment(date)
            value = income - spends
            months[f"{date.year}-{date.month}"] = {'income': income,
'spends': spends, 'value': value}
            date = get_1day_date(date - datetime.timedelta(days=25))

        translate = {
            'income': 'Прибуток',
            'spends': 'Витрати',
            'value': 'Чиста виручка'
        }

        diags = []
        for label, ua_label in translate.items():
            diags.append(
                go.Bar(x=[month for month in months], y=[data[label] for data
in months.values()], name=ua_label))
        fig = go.Figure(data=diags)
        graph_html = fig.to_html()

        group_dict = {'group': group, 'students': [], 'graph_html':
graph_html}
        students = group.students.all()
        for student in students:
            userprofile = student.company_user.userprofile
            group_dict['students'].append({'student': student, 'userprofile':
userprofile})
        result.append(group_dict)

    return render(request, 'groups.html', context={'groups': result,
'comp_user': comp_user})
```

Тут так само збираються дані за останні 6 місяців, тільки цього разу дані складаються в більш складну структуру даних, щоб диференціювати їх по групах.

Після чого так само генерується html-шаблон, і ми бачимо діаграму на екрані.

Один із основних алгоритмів лежить в моделях.

```
def calculate_students_payment(self, month: datetime.date=None):
    if month:
        lessons = Lesson.objects.filter(
```

```
            date_start__gte=datetime.date(month.year, month.month, 1),
            date_start__lte=datetime.date(month.year, month.month,
calendar.monthrange(month.year, month.month)[1]),
            group=self
        )
    else:
        lessons = Lesson.objects.filter(group=self)
    return sum([lesson.calc_student_payment() *
lesson.present_students.count() for lesson in lessons])
```

Це – функція для підрахунку оплат студентів по групі.

Вибираємо межові дати, дістаємо дані всіх уроків, і підраховуємо їх за раніше виведеною формулою:

```
def calc_student_payment(self):
    return self.group.course.student_payment * (1 /
(self.group.students.count() ** 0.5))
```

Так само реалізований і падрахунок зарплатні для викладача, з невеликою модифікацією у формулі:

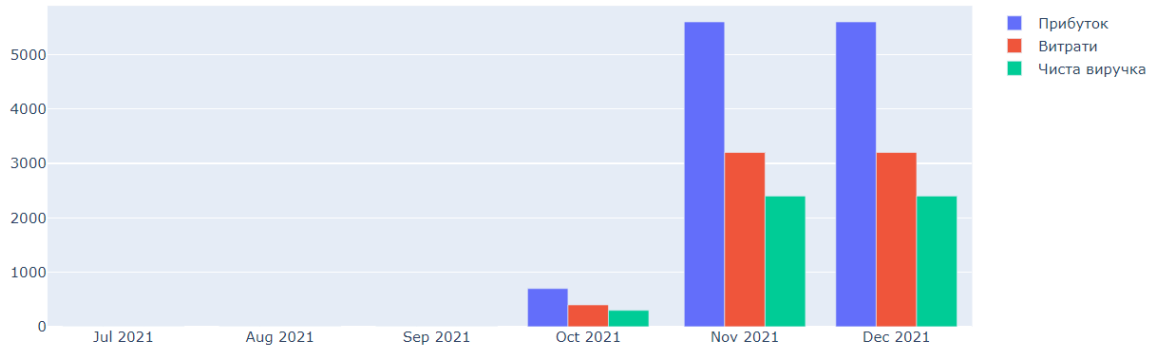
```
def calculate_teacher_payment(self, month: datetime.date=None):
    if month:
        lessons = Lesson.objects.filter(
            date_start__gte=datetime.date(month.year, month.month, 1),
            date_start__lte=datetime.date(month.year, month.month,
calendar.monthrange(month.year, month.month)[1]),
            group=self
        )
    else:
        lessons = Lesson.objects.filter(group=self)
    return sum([lesson.calc_teacher_payment() for lesson in lessons])
```

```
def calc_teacher_payment(self):
    return self.group.course.teacher_payment * (self.present_students.count()
** 0.5)
```

Тестові дані

В базі даних був створений необхідний набір даних для тесту, щоб не довелося нічого створювати власноруч.

Тож на зображенні бачимо саме один із тестових прикладів з даними:



Ім'я	Прізвище	Середня оцінка	Приєднався
Анна	Василенко		Dec. 27, 2021, 7:45 p.m.
Роман	Васько		Dec. 27, 2021, 7:46 p.m.
Ілля	Меленюк		Dec. 27, 2021, 7:46 p.m.
Катерина	Ігнатенко		Dec. 27, 2021, 7:46 p.m.

Був створений тестовий набір студентів, курсів, предметів та основна компанія «Матан».

Всі дані зберігаються в файлі базі даних sqlite3, який лежить у репозиторії, тож при локальній розгортці сайту вся ця інформація залишиться.

Висновки

В результаті отримали сайт з можливістю входити в компанії та створювати свої (це можна легко зробити в адмін-панелі).

Було змодельовано систему збереження даних користувачів та інфраструктури освітніх компаній, а також функціонал для автоматичного прорахунку оптимальних цін для клієнтів та оплат викладачам.

Окрім того, був реалізований аналіз отриманих даних, результати якого зображаються у вигляді таблиць та діаграм.

В майбутньому сайт можна і потрібно розвинути, проробивши усі CRUD – операції (створення, редагування та видалення даних різних таблиць) окремо від адмін-панелі, а також реалізувавши повноцінне меню для студентів, щоб ті мали змогу отримувати домашні завдання та виконувати їх. Тим не менш, зараз вже реалізована система прорахунку оптимальних цін для компаній. Окрім того, що дуже важливо, розроблена схема бази даних із усіма внутрішними зв’язками, що дозволяє автономно створювати інфраструктуру компанії із усіма предметами, курсами, групами, студентами та викладачами.

Література

1. Lightweight Django (Mark Lavin)
2. Django. Розробка веб-застосунків на Python (Уеслі Чан)
3. Design Patterns: Elements of Reusable Object-Oriented Software (Erich GammaRichard HelmRalph JohnsonJohn Vlissides)
4. Wikipedia [електронний ресурс]: https://ru.wikipedia.org/wiki.
5. METANIT [електронний ресурс]: http://metanit.com.
6. Django документація [електронний ресурс]: https://docs.djangoproject.com/en/4.0/