МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХЕРСОНСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
Факультет комп'ютерних наук, фізики та математики
Кафедра інформатики, програмної інженерії та економічної кібернетики

РОЗРОБЛЕННЯ БАЗИ ДАНИХ ДЛЯ СЕРВІСУ «ROZCLOUD»

Кваліфікаційна робота (проєкт)
на здобуття ступеня вищої освіти «бакалавр»

Виконав: студент 4 курсу
Спеціальності: 122 Комп'ютерні науки та інформаційні технології
Освітньо-професійної програми «Комп'ютерні науки та інформаційні технології» першого (бакалаврського) рівня освіти
Заобурний Владислав Вадимович
Керівники: старший викладач Черненко Ірина Євгенівна, кандидат фізико-математичних наук, доцент Єрмолаєв Вадим Анатолійович
Рецензент: кандидат педагогічних наук, доцент Таточенко Володимир Іванович

Херсон – 2020

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД	База даних
ПК	Персональний комп’ютер
СУБД	Система управління базами даних
API	Application Programming Interface
FTP	File Transfer Protocol
HTTP	HyperText Transfer Protocol
IP	Internet Protocol
JSON	JavaScript Object Notation
ORM	Object-Relational Mapping
SQL	Structured Query Language
XML	Extensible Markup Language

ВСТУП

Актуальність теми. На сьогоднішній день багато процесів та явищ в нашому житті переходять у інформаційний простір. Люди спілкуються в соціальних мережах, працюють за комп'ютерами та проводять вільний час за переглядом серіалів або відеоіграми. Сьогодні мабуть вже в кожного є смартфон або ноутбук. Саме тому діджиталізація всіх сфер людської діяльності є лише питанням часу. Не є виключенням і освіта. У нашій державі, а також в інших країнах колишнього радянського союзу весь освітній процес вже перейшов у інформаційний простір, а в деяких вишах – навіть у смартфони. І ця тенденція є зрозумілою, адже набагато зручніше та надійніше зосередити весь освітній процес у своєму смартфоні чи ПК.

Виходячи з цього є потреба в продукті, що зможе полегшити життя студентам, викладачам та іншим співробітникам вищих навчальних закладів. Розклад занять, налагодження комунікації між студентами та викладачами, розповсюдження навчальних матеріалів, новини факультету та університету та багато іншого буде зосереджено в цій системі, яка структурно поділяється на:

- Клієнт-частину – мобільний додаток для абітурієнтів студентів, викладачів та випускників. За допомогою додатку користувач може переглядати свій розклад, своєчасно дізнаватися про зміни в ньому, отримувати доступ до навчальних матеріалів, дізнаватися про новини свого вишу та багато іншого.
- Адмін-частину – веб-додаток, в якому працівники кафедри/ факультету/університету зможуть легко складати розклад, швидко та своєчасно доносити інформацію до студентів та викладачів, ділитися новинам кафедри/факультету/університету та інше.

- Сервер-частину – база даних та методи її передачі для зв’язку між клієнт- та адмін-частинами. В базі даних буде зберігатись весь архів інформації про всю роботу системи, що надасть змогу швидко знаходити всю інформацію щодо навчального процесу та робити глибокий аналіз методики навчання.

Розроблення такого масштабного сервісу доцільніше описати всі вимоги до функціоналу сервісу та, визначивши необхідний мінімум для його запуску, розбити продукт на модулі, які будуть поступово впроваджуватись доповнюючи цей сервіс. Це дозволить досягнути при проектуванні архітектури та розробленні усіх частин максимальної гнучкості та здатності до легкого масштабування.

В дипломній роботі описані вимоги до сервісу та БД у цілому. Також показані проєктування і розроблення БД та написання для взаємодії з клієнт-частиною.

Мета дослідження – проєктування та розроблення бази даних для сервісу «Rozcloud».

Завдання дослідження:

1. Визначити вимоги до продукту.
2. Проаналізовати аналогічні сервіси.
3. Визначити вимоги до бази даних.
4. Визначити основні сутності.
5. Визначено атрибути сутностей.
6. Визначено зв’язки між сутностями.
7. Досліджено сучасні інструменти для розробки бази даних.
8. Розроблено базу даних продукту.

Об’єкт дослідження: реляційні бази даних.

Предмет дослідження: база даних сервісу «Rozcloud».

Практичне значення полягає в наданні швидкого доступу до персоналізованого розкладу студентам та викладачам вищих навчальних закладів.

Структура дослідження. Дипломна робота складається зі вступу, списку умовних скорочень, двох розділів, висновків, списку використаних джерел та додатків.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ВИЗНАЧЕННЯ ВИМОГ

1.1. Архітектура «клієнт-сервер»

Розроблення сервісу вимагає забезпечити зберігання великих об’ємів даних та швидку й персоналізовану передачу їх користувачам. Виходячи з цього стає очевидним використання клієнт-серверної архітектури.

Як зрозуміло з назви, в даній концепції беруть участь дві сторони: клієнт і сервер. Тут все як у житті: клієнт - це замовник тієї чи іншої послуги, а сервер - постачальник послуг. Клієнт і сервер в нашому випадку фізично представляють собою мобільний додаток та базу даних з набором програм для вибірки даних з цієї бази та отримання їх від користувача, їх обробки та занесення в базу чи відправки у клієнт-частину сервісу.

Клієнт і сервер взаємодію один з одним у мережі Інтернет або в будь-якій іншій комп'ютерної мережі за допомогою різних мережевих протоколів, наприклад, IP протокол, HTTP протокол, FTP та інші. Повідомлення, які посилають клієнти отримали назву «HTTP-запити». Запити мають спеціальні методи, які говорять серверам про те, як обробляти повідомлення. А повідомлення, які посилає сервер отримали назву «HTTP-відповіді», вони містять, крім потрібної інформації, ще й спеціальні коди стану, які дозволяють клієнту дізнатися, як сервер зрозумів його запит [2].

Також варто зауважити, що в основі взаємодії клієнт-сервер лежить принцип того, що така взаємодія починається клієнтом, сервер лише відповідає клієнту і повідомляє про те чи може він надати послугу клієнтові і якщо може, то на яких умовах. Клієнтське програмне забезпечення та серверне програмне забезпечення зазвичай встановлено

на різних пристроях, але також вони можуть працювати і на одному комп'ютері.

Дана концепція взаємодії була розроблена в першу чергу для того, щоб розділити навантаження між учасниками процесу обміну інформацією, а також для того, щоб розділити програмний код постачальника і замовника. (Рис. 1.1)

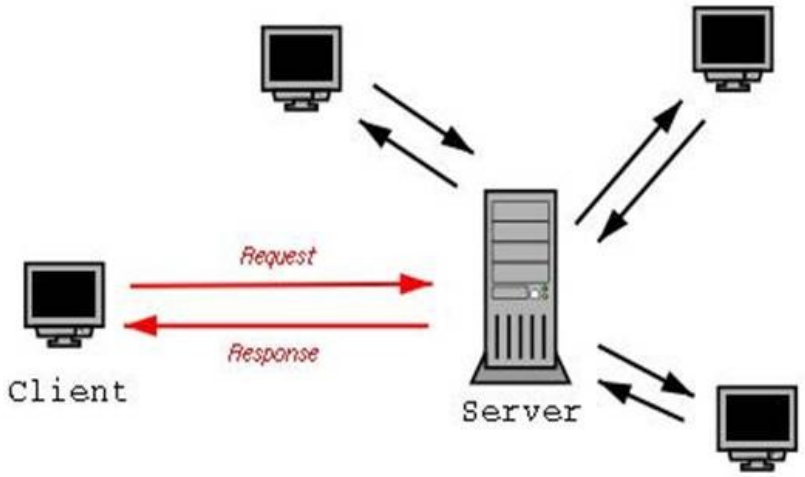

Рис. 1.1. Клієнт-серверна взаємодія

Ми бачимо, що до одного сервера може звертатися відразу кілька клієнтів. Також варто зауважити, що кількість клієнтів, які можуть одночасно взаємодіяти з сервером залежить від потужності сервера і від того, що хоче отримати клієнт від сервера.

Багато мережних протоколи побудовані на архітектурі клієнт-сервер, тому в їх основі зазвичай лежать однакові або схожі принципи взаємодії, а різницю ми бачимо лише в деталях, які обумовлені особливостями і специфікою області, для якої розроблявся той чи інший мережевий протокол [3].

1.2. Вимоги до сервісу

Перед тим як розпочати роботу над бази даних, ми визначили ряд головних вимог до продукту:

- Є декілька типів користувачів: студент, викладач та співробітник;
- Користувачі мають різний функціонал в залежності від типу свого облікового запису;
- Є декілька структур з ієрархічним функціоналом: кафедра, факультет, університет;
- Структури мають різний функціонал в залежності від типу;
- Структура об’єднує під собою «співробітників»;
- Реєстрація користувачів відбувається користувачами з вищої ланки ієрархії: співробітник університету –> співробітник факультету –> співробітник кафедри –> викладач/студент;
- Складання розкладу відбувається в адмін-частині співробітниками;
- Користувач авторизується за наданим йому логіном та паролем;
- Студент/викладач може переглядати свій розклад за вибраним тижнем/місяцем;

На основі цих вимог до альфа-версії сервісу буде здійснюватися проєктування бази даних, а також визначатися список ключових критеріїв для аналізу аналогів.

Окрім цього під час аналізу буде звертатись увага на структуру сервісу, доцільність інформації, що надається та типи користувачів.

1.3. Аналіз аналогів

Перед тим як розпочати роботу над будь-яким продуктом треба ознайомитись з аналогами, що вже існують. Це дасть змогу побачити всі переваги та недоліки тих чи інших рішень, а також зрозуміти потреби користувачів. Не зважаючи на те що ознайомитись зі структурою БД не вдасться, одразу стане зрозуміло як влаштована передача даних у сервісі та наскільки широкий и персоналізований функціонал надається користувачу.

Для аналізу були обрані додатки в Play Market, що мають найбільшу кількість скачувань:

- «Расписание занятий МКР» (https://play.google.com/store/apps/details?id=com.mkr.shedule_app)
  У додатку є багато вишів. Структура системи аналогічна нашому продукту. Широкий функціонал.
- «Расписание занятий для школы и вузов – Weeklie» (https://play.google.com/store/apps/details?id=com.numen.timetable)
  Влучний інтерфейс. Розклад заповнюється користувачем.
- «СтудЖурнал - Расписание занятий» (https://play.google.com/store/apps/details?id=com.romansytnyk.studentstudio)
  Непогана структура додатку. Розклад заповнюється користувачем.

У всіх додатках такого формату було виявлено багато недоліків с точки зору ролей в сервісі. Масштабованість сервісів не є можливою. Оцінки від користувачів не перевищували 4 зірки. Загалом вибрані аналоги лише частково підходять для аналізу лише за деяким функціоналом. Сервіс «Rozcloud», а отже і структура БД та API передбачають зовсім інший підхід до реалізації серверної частини.

Після проведення аналізу були виявлені основні недоліки, на які було звернено увагу.

«Расписание занятий МКР»:

- Відсутність авторизації – користувач може лише переглядати розклад у режимі гостю. Проте є централізована БД для кожного університету.
- Через відсутність авторизації користувачів стає неможливим архівація даних для подальшого аналізу.
- Через відсутність авторизації стає неможливою будь-яка взаємодія між користувачами.
- Дані студентів та викладачів у відкритому доступі.
- Відсутні налаштування персоналізації та конфіденційності.

«Расписание занятий для школы и вузов – Weeklie»:

- Розклад заповнюється користувачем;
- Відсутня авторизація та персоналізація;

«СтудЖурнал - Расписание занятий»:

- Розклад заповнюється користувачем;
- Відсутня авторизація та персоналізація;
- Функціонал обмежений лише переглядом розкладу та заповненням домашнього завдання.

1.4. Визначення вимог до серверної частини

На основі вимог до продукту в цілому та аналізу аналогів були виведені основні вимоги до бази даних:

- Масштабованість – можливість додавання нових сутностей в рівнях користувача та структур;
- Доцільне за об’ємами зберігання інформації;
- Закритість – доступ до даних мають лише авторизовані користувачі;
- Гнучкість – через додавання нових модулів в наступних версіях база даних не повинна перероблятись
- Захищеність – використання методів криптографії для обміну інформацією між клієнт/адмін-чатсиною та базою даних;
- Фіксація зміни/заміни інформації в базі даних та уникнення втрати старої інформації;
- Уникнення розповсюдження (та оперування) персональними даними користувачів;
- Можливість авторизації для викладача та студента;
- Можливість переглядати персоналізований розклад в залежності від типу користувача та його належності до тих чи інших структур;
- Наявність відповідної БД для створення екземплярів сутностей користувачів, структур, розкладу.

На основі визначених вимог відбувались проєктування та розроблення бази даних.

1.5. Вибір інструментів

СУБД

Для того щоб розробити серверну частину нам потрібно обрати фреймворк та СУБД.

Так як «Rozcluod» в подальшому буде оперувати великим об’ємами багато структурної інформації, серед усіх інших мною були обрані дві найпотужніші СУБД – PostgreSQL та Neo4j.

***PostgreSQL*:**

Переваги:

- Найшвидша реляційна СУБД;
- Строга типізація та структурність даних;
- Дає змогу компактної структуризації даних.

Недоліки:

- Поступається в швидкодії нереляційним СУБД при великих об’ємах даних.

***Neo4j*:**

Переваги:

- Забезпечує високу швидкодію навіть при великих об’ємах даних;
- Дозволяє створювати зв’язок у вигляді об’єктів;

Недоліки:

- Мала швидкодія при малих об’ємах даних порівняно з реляційною БД;
- Складний в освоєні;
- Не строго структурована інформація.

З огляду на зазначені вище переваги було обрано СУБД PostgreSQL [9].

Фреймворк

Фреймворк надає інструменти та бібліотеки, які спрощують завдання серверної розробки, включаючи взаємодію з базами даних, підтримку сеансів і авторизацію користувачів, форматування виводу (наприклад, HTML, JSON, XML) і поліпшення захисту від веб-атаки.

Вибір одразу пав на Django, так як в нього є багато бібліотек, що спрощують написання API в тому числі і при взаємодії з PostgreSQL. Окрім цього я вже знайомий з мовою Python на якій написаний Django, що позитивно вплине на швидкості та якості розробки [1].

Також можна виділити наступні переваги Django як фреймворку:

- Більшість інструментів для створення програми - частина фреймворку, а не поставляються у вигляді окремих бібліотек.
- Містить величезну кількість функціональності для вирішення більшості завдань веб-розробки (ORM, міграції бази даних, автентифікація користувача, панель Адміністратора, форми)
- Задає структуру проекту. Вона допомагає розробнику розуміти, де і як додавати нову функціональність. Завдяки цьому набагато простіше знайти вже готові рішення або отримати допомогу від спільноти.
- Додатки в Django, які дозволяють розділити проект на декілька частин та дозволяє легко інтегрувати готові рішення.
- Безпечний за замовчуванням і включає механізми запобігання поширених атак на зразок SQL-ін'єкцій і підробки міжсайтових запитів.

РОЗДІЛ 2
ПРОЄКТУВАННЯ ТА РОЗРОБЛЕННЯ БАЗИ ДАНИХ

Неможливо створити БД без докладного її опису, також як і не можливо зробити будь-який складний виріб без креслення і докладного опису технологій його виготовлення. Іншими словами, потрібен проект. Проектом прийнято вважати ескіз деякого пристрою, який в подальшому буде втілений в реальність. Процес проєктування БД являє собою процес переходів від неформального словесного опису інформаційної структури предметної області до формалізованого опису об'єктів предметної області в термінах деякої моделі. Кінцевою метою проєктування є побудова конкретної БД.

Збір відомостей і системний аналіз предметної області - це перший і найважливіший етап при проектуванні БД. У ньому необхідно провести докладний словесний опис об'єктів предметної області і реальних зв'язків, присутніх між реальними об'єктами. Основою аналізу коректності схеми є функціональні залежності між атрибутами БД. У деяких випадках між атрибутами відносин можуть з'явитися небажані залежності, які викликають побічні ефекти і аномалії при модифікації БД. Для ліквідації можливих аномалій передбачається проведення нормалізації відносин БД [6].

Виокремлення сутностей та визначення зв’язків між ними – це основна задача при проектуванні. Тип залежності однієї сутності від іншої, деталізація атрибутів сутностей, винесення спільних атрибутів у сутність більш абстрактного рівня та інші методи нормалізації - все це обов’язковими умовами для правильного проєктування бази даних для будь-якого продукту.

2.1. Визначення сутностей

Для того щоб почати роботу над проєктуванням треба спочатку виділити сутності системи. В сервісі «Rozcloud» наразі надається можливість лише для переглядання розкладу.

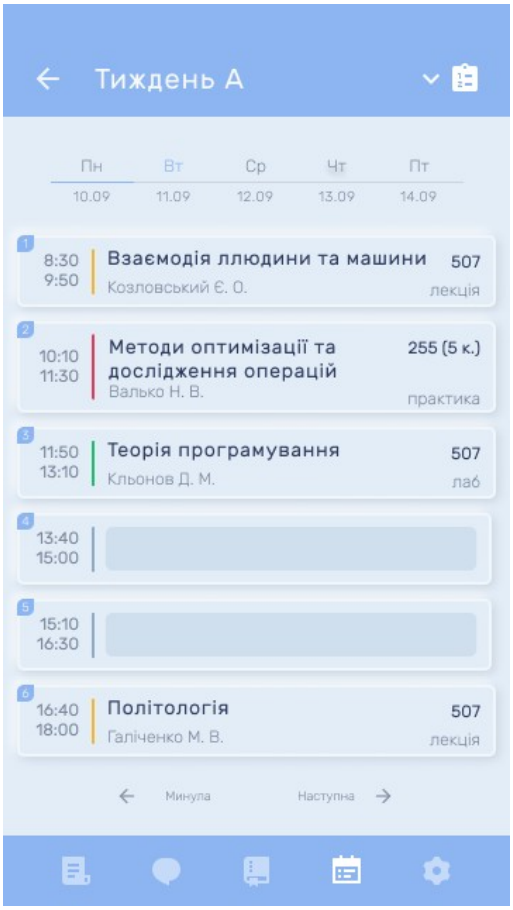

Рис. 2.1.1. Тижневий розклад

З цього екрану мобільного додатку (Рис. 2.1.1.) можна одразу визначити деякі сутності. По перше в є пара. У пари є предмет, викладач, аудиторія, тип пари номер та час. Це те що видно одразу. Також очевидно, що пари знаходяться в дні, а дні в тижні.

Отже, якщо заглибитись у сутність розкладу, то ми познайомимось з його механізмами складання, які відбуваються у реальному житті в стінах університету. Так ми дізнаємося, що тижневий розклад складається кожний модуль на основі графіку освітнього процесу, який

в свою чергу розробляється на основі робочого плану. А робочий план має бути узгоджений з навчальним планом, який приймають кожен рік для першокурсників на всі 4 роки навчання (якщо говорити про бакалаврів). Таким чином нам спочатку треба ознайомитись з повним процесом складання усіх цих планів, щоб правильно спроектувати БД, та в подальшому надати змогу співробітникам факультетів і кафедр складати ці плани в адмін-частині нашого сервісу.

Окрім навчальних дисциплін та їх структури, можна побачити викладачів. А приведений розклад (Рис. 2.1.1.) відображається для студента. В подальшому окрім цих типів користувачів ще з’явиться співробітник. Все це є окремими сутностями які вже можна спостерігати.

А також і розклад і студенти з викладачами мають належність до своїх кафедр, факультетів, університетів. Всі ці структури теж слід виділити як сутності. Якщо міркувати далі, то в університету можуть бути корпуси в яких є аудиторії. Корпуси і аудиторії можуть бути закріплені за кафедрами чи факультетами. Це теж слід мати на увазі при проектуванні БД.

Первинний ключ в сутностях завжди буде автоінкрементним цілим числом. Також обов’язкові поля з датою і часом створення екземпляру сутності та датою і часом його останнього оновлення. Це потрібно для більш простого та оптимального спілкування з клієнт-частиною.

В цілому маємо користувачів, структури, різні сутності для розкладу та деякі окремі сутності для інших даних. Тепер можна перейти до проєктування конкретних сутностей і зв’язків між ними.

2.2. Структури

Оскільки сервіс «Rozcloud» розробляється для вищих навчальних закладів, то й багато сутностей в ньому будуть мати зв’язок з університетами до яких вони відносяться. Саме тому буде доцільним почати проєктування БД з сутності університету та її зв’язку з іншими сутностями.

Університет

university	
id	int
created_at	timestamp
updated_at	timestamp
name	varchar(128)
reduction	varchar(8)
logo	bytea
area	varchar(32)
city	varchar(128)
adress	varchar(255)

Рис. 2.2.1. Таблиця «university»

Окрім стандартних атрибутів – первинний ключ (id), дата створення (created_at), дата останнього оновлення (update_at) – сутність «Університет» має назву, абревіатуру, логотип, та місце знаходження (область, місто, адресу). Назва університету буде заповнюватись в адмін частині, а абревіатура буде автоматично збиратись та заповнюватись з назви. Логотип університету - поле яке буде містити посилання на файл, що знаходиться на сервері. Область та місто поки що будуть реалізовані у вигляді enum. Це масив із варіантів значень цього поля. В першій версії сервісу в нас буде лише одна область та одне місто, тому задля

економії пам’яті я обрав таку реалізацію. В подальшому будуть створені окремі сутності для міста та області, а університет буде зв’язаний з ними зовнішнім ключем. Адреса буде залишатись текстовим полем, яке заповнюється з адмін-панелі, так як зберігати всі вулиці міста поки що є недоцільним у випадку нашого сервісу.

Корпус

В багатьох університетах є декілька корпусів і зазвичай кожен студент денної форми навчання таких університетів хоча б раз мав пари в різних корпусах. Що ж таке корпус, якими атрибутами він володіє і який в нього сенс у нашому сервісі?

В кожного вищого навчального є перший корпус – головний. В одному корпусі можуть розміщуватися всі факультети вишу, проте в університетах часто одного корпусу не вистачає на всі факультети, тому що в університетах їх в середньому близько п’ятнадцяти. Тому деякі факультети знаходяться в інших корпусах (другому, третьому і т.д.). Також деякі корпуси можуть створюватись спеціально для лабораторних робіт і мати спеціальні аудиторії з відповідним оснащенням.

Якщо студент навчається наприклад на факультеті комп’ютерних наук, який знаходиться в головному корпусі і в нього є пара що проходить в 5-ому корпусі в 205-ій аудиторії – в розкладі буде вказана не тільки аудиторія, а ще й корпус, тому що за замовчування аудиторія без вказання корпусу мається на увазі саме в корпусі факультету (кафедри) де навчається студент (в нашому випадку в головному).

Саме тому важливо виділити корпус як окрему сутність. Про його зв’язок з аудиторіями та факультетами буде розказано в наступних пунктах.

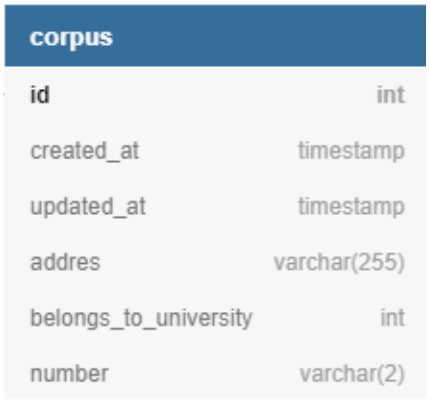

Рис. 2.2.2 Таблиця «corpus»

Корпус буде обов’язково мати адресу, що буде текстовим полем, номер та належність до університету, що є зовнішнім ключем зі зв’язком типу «один до багатьох». Без заповнення всіх цих полів екземпляр цієї сутності не має бути створений, так як не буде містити інформації задля якої створюється.

Факультет

Факультет собою уявляє сутність не фізичної структури, проте академічно важливої так як студенти, викладачі і співробітники належать до певних кафедр, які в свою чергу належать до відповідних факультетів.

Факультет має назву, абревіатуру (заповнюється автоматично в залежності від назви), логотип (поле уявляє собою рядок з посиланням на файл, що знаходиться на сервері) та належність до корпусу, що є зовнішнім ключем з типом зв’язку «один до багатьох».

Поки сервісом буде користуватись один університет, нам не потрібно поле належності до університету, так як вибірки ми будемо робити лише за одним єдиним університетом. Проте ми зможемо в будь-яку мить додати це поле да дуже швидко заповнити його для всіх екземплярів цієї сутності.

faculty	
id	int
created_at	timestamp
updated_at	timestamp
belongs_to_corpus	int
name	varchar(255)
reduction	varchar(8)
logo	bytea

Рис. 2.2.3 Таблиця «faculty»

Кафедра

Кафедри являються ключовою сутністю що зв'язує студентів, викладачів та співробітників з їхніми факультетами та вищими навчальними закладами.

Атрибути кафедри аналогічні атрибутам факультету, проте на відміну від факультету кафедра належить не до корпусу, а до факультету. Не дивлячись на ідентичність атрибутів цих двох сутностей (кафедри та факультету) виносити спільну частину в окрему сутність вищої абстракції в цьому випадку є зайвим.

department	
id	int
created_at	timestamp
updated_at	timestamp
belongs_to	int
name	varchar(255)
reduction	varchar(8)
logo	bytea

Рис. 2.2.4 Таблиця «department»

Аудиторія

Аудиторія – сутність яка напряму фігурує в розкладі в будь-яких випадках. В неї є належність до кафедри через зв’язок зовнішнім ключем, тип зв’язку «один до багатьох». Так було вирішено зробити для чіткого розрізнення зони відповідальності за аудиторії між кафедрами.

Через зв’язок кафедра-факультет-корпус за необхідністю можна дізнатись до якого корпусу належить аудиторія. Створювати окремий зовнішній ключ з належністю до корпусу на цьому рівні роботи сервісу не має потреби, так як кількість запитів до бази даних буде незначною.

Поля «номер» та «тип» є обов’язковими. Поле «поверх» буде заповнюватись самостійно по першій цифрі номеру аудиторії. Це поле створене для того щоб не опрацьовувати кожен раз номер аудиторії, якщо на буде потрібно зробити вибірку усіх аудиторій поверху. Поле «тип» поки що буде зовнішнім ключем до окремої сутності.

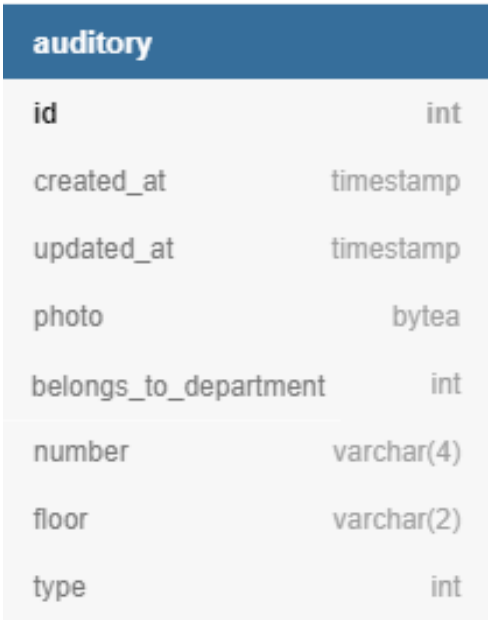

Рис. 2.2.5. Таблиця «auditory»

Так зроблено для того щоб університет сам вирішував які типи аудиторій в нього є (комп’ютерні класи, лекційні, лабораторії і т.д.). Ця сутність буде складатись з стандартних полів та поля назви. Поле

«Фото» є необов’язковим, та й як усі поля графічної інформації буде містити посилання на зображення, що зберігається на сервері.

На цьому опис структур та зв’язків між ними в нашій базі даних завершений. Далі переходимо до опису користувачів та їх зв’язків зі структурами.

2.3. Користувачі

На відміну від структур всі типи користувачів мають багато спільних даних, які можна винести в окрему сутність більш вищого рівню абстракції. Це дозволить спростити більшу кількість запитів до бази даних та зробить структуру більш оптимальною. Тому доцільно буде створити сутність «Користувач», на яку будуть посилатися всі інші сутності різних типів користувачів.

Користувач

Для того щоб визначити що з себе буде уявляти ця сутність, потрібно зрозуміти які атрибути є і в студента, і у викладача і в співробітника. Очевидними є такі атрибути як логін, пароль, хеш сесії та ПІБ та дата народження. Також в кожного з користувачів мають бути такі необов’язкові атрибути як номер телефону, електронна скринька та адреса проживання.

Логін та пароль в нашому сервісі для кожного користувача генеруються випадковим чином при реєстрації. В подальшому після авторизації він зможе їх змінити. Хеш сесії створюється та заноситься в базу даних при кожній зміні пароля та/або логіна, а також при кожній авторизації.

Так як всі зареєстровані користувачі в нашому сервісу є відомими співробітникам реальними людьми, дата народження прізвище ім’я та по батькові теж заповнюються з адмін-панелі при реєстрації. Ці поля є обов’язковими для заповнення під час реєстрації користувача. Без заповнення цих полів екземпляр сутності «користувач» не створиться.

Наразі в нашому сервісі не буде можливості заповнити необов’язкові поля «фото», «адреса», «телефон», «електронна скринька». Але вони плануються стани доступними для заповнення під час наступного оновлення сервісу. У сутності користувач немає ніяких зв’язків зі структурами, так як вони передбачені у сутностей типів користувачів, які будуть посилатися на дану.

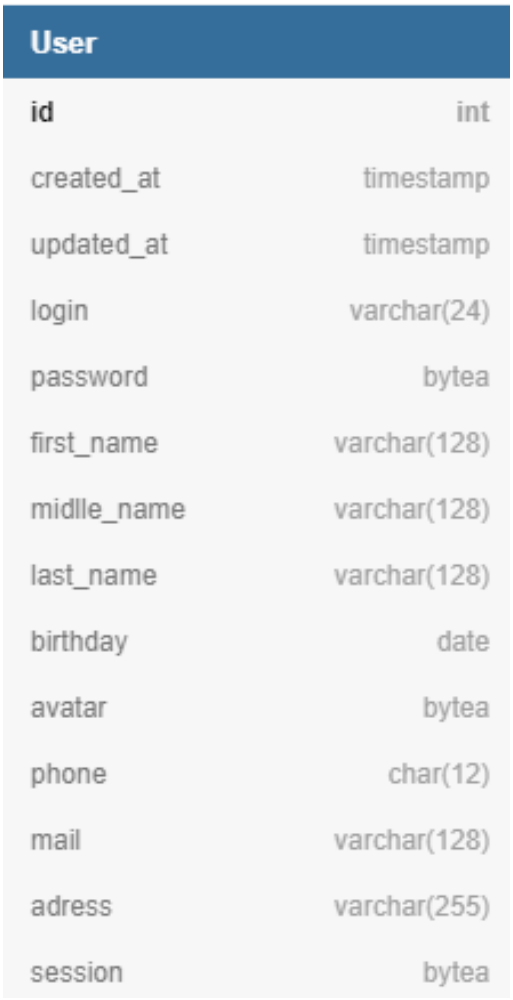

Рис. 2.3.1. «User»

Співробітник

Співробітники наразі не володіють характерними для них атрибутами, проте мають належність до відповідної структури. Співробітник в реальності це людина, яка працює на кафедрі, на факультеті чи в університеті. Вона може виконувати різні функції: складати розклад, займатися навантаженням викладачів та студентів,

спілкуватися з учасниками навчального процесу та надавати їм довідкову інформацію, тощо.

В сервісі «Rozcloud» наразі це користувач, який може реєструвати інших користувачів та складати розклад. В подальшому планується додавати ще багато функціоналу цьому типу користувачів, але поки що їх сутність можна обмежити лише належністю до відповідної структури (зв'язок типу «один до багатьох») та належністю до сутності користувача (зв'язок типу «один до одного»).

Проте неможливо не зазначити що співробітники різних структур в подальшому будуть мати й різний функціонал та різні права доступу. Навіть зараз реєструвати користувачів типу «студент» можуть всі співробітники, проте реєстрація користувачів типу «викладач» може здійснюватися лише співробітниками факультету та університету. Поля «створення» та «оновлення» тут непотрібні, так як ці сутності посилаються на сутність «користувач», де ці поля існують, а у власне цих таблицях відсутня будь-яка окрім зовнішніх ключів.

emploee_university		emploee_faculty		emploee_department	
id	int	id	int	id	int
belongs_to_university	int	belongs_to_faculty	int	belongs_to_department	int
belongs_to_user	int	belongs_to_user	int	belongs_to_user	int

Рис. 2.3.2. «emploee»

У реальному житті, звичайно, рішення про вступ студентів та прийом на роботу викладачів приймаються на рівні університету, проте в сервісі зроблено так для більш оптимального розподілення роботи. В подальшому планується зробити підтвердження з боку працівника університету щодо прийняття таких рішень, в той час як робота по заповненню даних та реєстрації залишиться на працівниках факультетів та кафедр.

Реєстрація ж самих співробітників здійснюється лише співробітниками що належать до більш вищих структур. Таким чином співробітників кафедри можуть реєструвати лише співробітники факультету та університету, співробітників факультету – лише співробітники університету. Останні в свою чергу реєструються адміністраторами сервісу. Без прив'язки до структури екземпляр сутності співробітника створений бути не може.

Викладач

Викладач наразі окрім належності до «користувача» має вчене звання, яке є полем вибору з константного масиву, так як всі варіанти є затвердженими на законодавчому рівні і навряд чи будуть змінюватись найближчим часом. Також у викладача є належність до кафедри на якій він викладає (зв'язок типу «один до багатьох»). Без прив'язки до кафедри екземпляр сутності викладача створений бути не може.

Рис. 2.3.3. «teacher»

Студент

Сутність студента окрім належності до «користувача» має необов'язкові поля для серії та номеру студентського квитка які наразі неможливо буде заповнити в адмін-панелі. В сервісі «Rozcloud» ми хочемо уникнути оперування персональними даними, так як це тягне за собою багато роботи з узгодженням питань на рівні законодавство с приводу оперування персональними даними. Наразі для першої версії

сервісу, яка створюється для перевірки доцільності сервісу та попиту серед учасників навчального процесу, це є недоцільним.

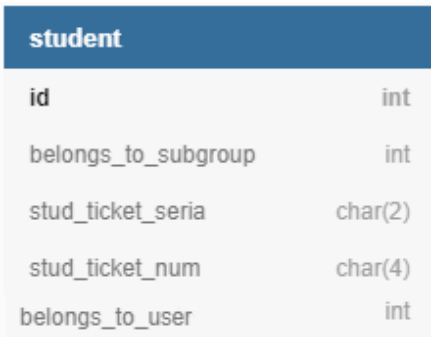

Рис. 2.3.4. «student»

Також на відміну від викладача та співробітника, студент має належність не до структури а до підгрупи, в якій навчається.

Група та підгрупа

Багато явищ та процесів вищих навчальних закладів, що стосуються студентів направлені не на кожного студента зокрема, але на групу студентів в цілому. Тестування, проектна діяльність та багато іншого мають спільний сенс для всіх студентів конкретної академічної групи. Та що наразі саме головне – це розклад, який складається у вишах не для кожного студента окремо, а для групи (в деяких випадках – для підгрупи) студентів.

З огляду на це доцільно буде створити дві сутності «група» та «підгрупа», так як часто буває що розклад складається по деяким дисциплінам саме для підгруп, проте багато запитів до бази даних в нашому сервісі будуть спрямовані саме на групу.

Підгрупа має номер («1» або «2») та належність до групи (зв’язок типу «один до багатьох»). Більше атрибутів їй не потрібно, адже ця сутність створена лише для персоналізації розкладу для користувачів.

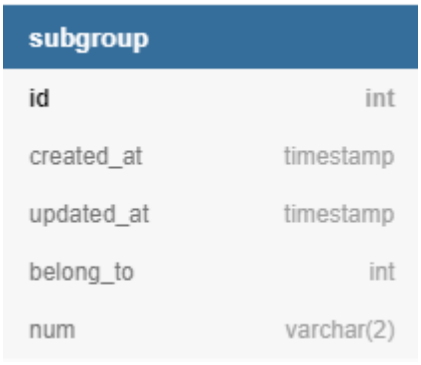

Рис. 2.3.5. «subgroup»

Що стосується групи – то вона вже має назву (номер), курс та належність до спеціальності. Таким чином студенти певної що вступили до вишу у 2020 році будуть мати сутність групи з атрибутами певної спеціальності, номеру групи та курсу. А відрізняти від наступного покоління їх буде дата створення екземпляру сутності їх групи. Всі поля сутності групи обов’язкові до заповнення.

Рис. 2.3.6. «group»

Поле «спеціальність» є зовнішнім ключем (зв’язок типу «один до багатьох») до однойменної сутності, яка грає важливу роль у структурі БД.

Спеціальність

Спеціальність це сутність яка об’єднує багато груп та зв’язує їх з кафедрою. У спеціальності є назва та шифр, а також належність до

кафедри (зв’язок типу «один до багатьох»). Всі поля цієї сутності є обов’язковими. Таким чином ми зв’язали студентів з кафедрою: «студент» -> «підгрупа» -> «група» -> «спеціальність» -> «кафедра».

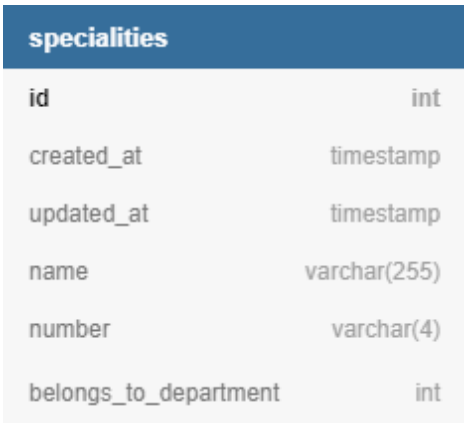

Рис. 2.3.7. «specialities»

2.4. Розклад

У вищих навчальних закладах розклад, який студенти отримують від своїх кафедр складається для груп (підгруп) на основі графіку освітнього процесу, який в свою чергу складається виходячи з робочого плану. А робочий план являє собою розподілений по семестрам (рокам) навчальний план.

В сервісі «Rozcloud» розклад буде складатися таким же чином, проте не в електронних таблицях, а в адмін-панелі зі зручним інтерфейсом. Наявність централізованої БД дозволить співробітникам фіксувати багато нюансів (перенавантаження викладачів та студентів, конфлікт аудиторій, тощо), що зекономить багато часу при складанні розкладу.

Навчальний план

Навчальний план складається для кожної групи студентів певної спеціальності на весь період навчання для здобуття кваліфікаційного рівня. Він уявляє собою таблицю з переліком дисциплін, які визначаються спеціальністю та розподілені на групи (цикл загальної

підготовки, цикл професійної підготовки тощо). В кожної з них є кількість годин та кредитів, що виділені на дисципліну і розподілені за типом (лекційні, практичні/семінарські, лабораторні та для самостійної роботи). Також ці години та кредити розподілені по семестрам і в кожному семестрі для дисципліни визначена форма контролю (залік, диференційований залік або екзамен).

Для того щоб зберігати план такого формату, знадобиться дві сутності: сутність власне навчального плану та сутність дисципліни. Це дозволить спростити запити до бази даних та в подальшому масштабувати функціонал зв’язаний з навчальними планами.

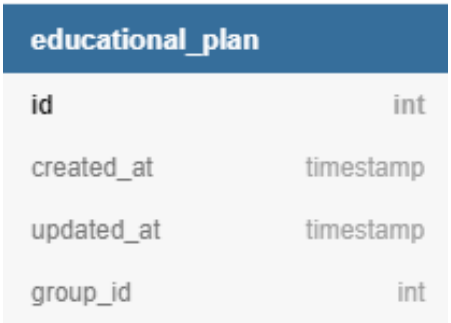

Рис. 2.4.1. «education_plan»

Сутність навчального плану наразі має лише належність до групи для якої складається цей план (тип зв’язку «один до одного»). В подальшому у цієї сутності можуть з’явитися інші артибути.

А сутність дисципліні в свою чергу буде зв’язана з навчальним планом зовнішнім ключем (тип зв’язку «один до багатьох»).

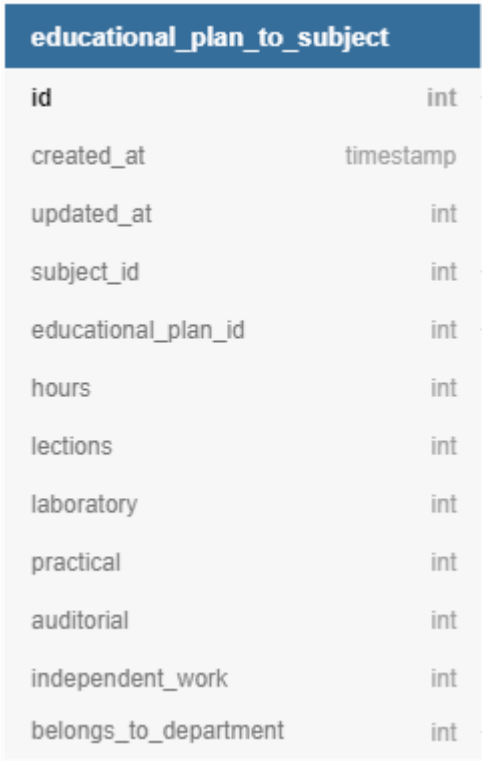

Рис. 2.4.2. «education_plan_to_subject»

Сутність дисципліні має зовнішній ключ до предмету (тип зв’язку «один до багатьох») та поля для різних типів годин. Години що зазначені в цій сутності є годинами виділеними на дисципліну на весь період викладання дисципліни. Також курс викладання кожної дисципліни розробляється певною кафедрою і читають цю дисципліну викладачі цієї кафедри. Тому в цій сутності є зовнішній ключ тип зв’язку «один до багатьох») до кафедри.

Сутність предмету бути мати лише поле назви.

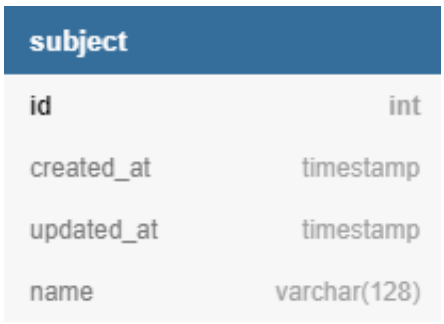

Рис. 2.4.3. «subject»

Окрім цього нам знадобиться сутність для типів контролю у семестрах, в яких викладається дисципліна.

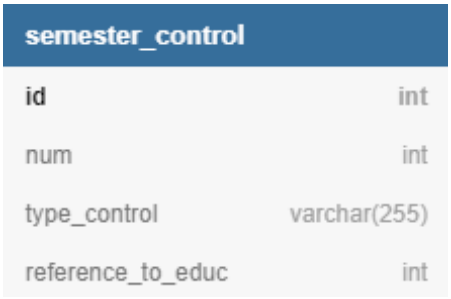

Рис. 2.4.4. «semester_control»

В цій сутності полями будуть: тип контролю (поле переліку), номер семестру та зовнішній ключ до сутності дисципліни. Таким чином при заповнюванні навчального плану на кожну дисципліну будуть створені екземпляри сутності тип контролю, що в подальшому дасть змогу дізнатись з навчального плану скільки семестрів та в яких саме має викладатися певна дисципліна та її форму контролю.

Робочий план

Робочий план за структурою буде подібний до навчального плану. Він теж буде розподілений на дві сутності. Перша сутність – власне робочий план, яка буде мати зовнішній ключ до навчального плану (тип зв’язку «один до багатьох») та семестр, на який складається план.

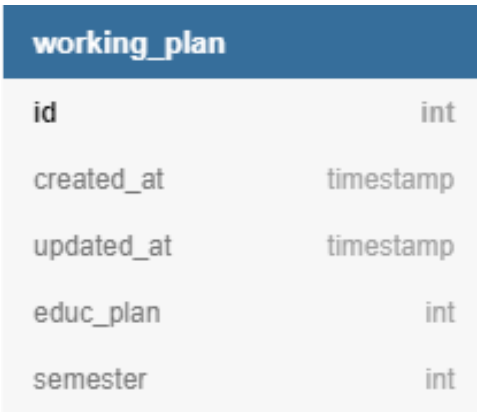

Рис. 2.4.5. «working_plan»

Друга сутність являє собою аналог дисципліни до якої має зовнішній ключ (тип зв'язку «один до багатьох») та посилається на робочий план (тип зв'язку «один до багатьох»).

Окрім годин викладання дисципліни в даному семестрі ця сутність має зовнішні ключі до викладачів для кожного типу занять (тип зв'язку «один до багатьох»), адже які саме викладачі що будуть проводити ті чі інші заняття даної дисципліни певній групі студентів визначається саме при складанні робочого плану.

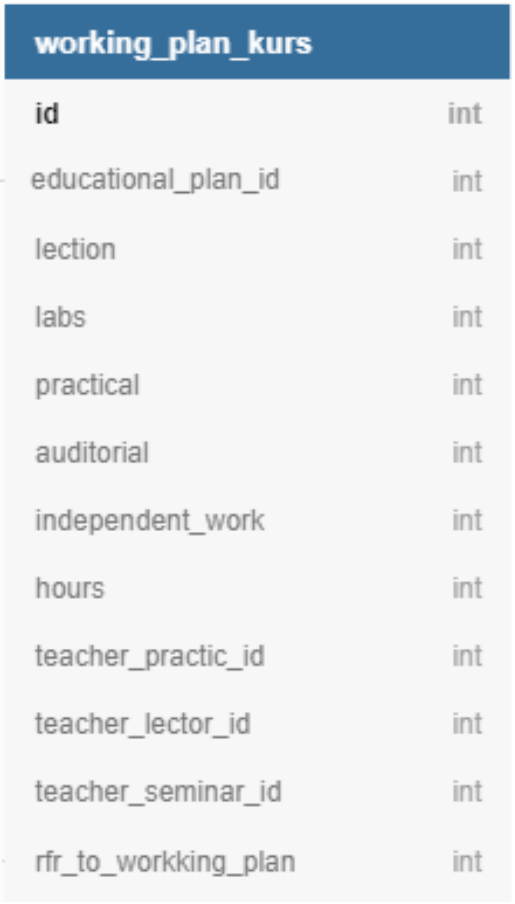

Рис. 2.4.6. «working_plan_kurs»

Також виносячи в цю таблицю дані про дисципліну на семестр ми маємо змогу в подальшому не дублювати їх, а використовувати для різних робочих планів одні й ті ж екземпляри, якщо це знадобиться. Те ж саме стосується й таблиці дисципліни при складанні навчального плану.

Тижневий план

Тижневий план в структурі бази даних є вираженням графіку освітнього процесу, адже його можна виразити у вигляді екземплярів тижневого розкладу. При створенні графіку освітнього процесу в кожному тижні вказується кількість годин типу занять усіх дисциплін.

Почнемо з того що кожний предмет має кількість годин в тижні, яка визначається в залежності від загальної кількості наданих на семестр годин. Зрозуміло що предмет має різні типи занять (пар), на які виділена різна кількість годин. Таким чином на предмет може бути виділено 60 годин, 40 з яких лекційні, а 20 – лабораторні. Для зберіганння цих даних ми створимо сутність «години предмету на тиждень».

Атрибути цієї сутності - це тип занять, кількість годин виділений на цей тип та зовнішній ключ до дисципліни робочого плану для якої вона складається. Такий зв'язок дозволить нам оперувати при створенні графіку освітнього процесу лише предметами з відповідного робочого плану. Іншими словами цей зовнішній ключ розповідає нам про кількість годин на тиждень конкретного типу занять конкретної дисципліни.

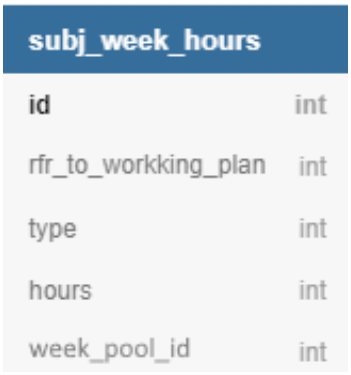

Рис. 2.4.7. «subj_week_hours»

Цей набір екземплярів сутності буде дублюватись для різних тижнів, адже більшу частину навчального процесу складає два тижні «А» і «Б» які чергуються один за одним. Проте кількість годин занять

повинна складатись на кожний навчальний тиждень, який має свій порядковий номер та дати. Саме тому створимо таблицю-посередника, яка допоможе реалізувати такий зв’язок (тип «багато до багатьох»).

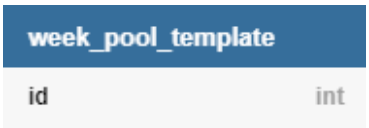

Рис. 2.4.8. «week_pool_template»

І лише тепер створюється сутність «тиждень», до якого прив’язуємо набір кількості годин виділених на кожний тип заняття кожної дисципліни.

Окрім вище описаного зв’язку ця сутність має порядковий номер, тип, опис та дату початку, яка знадобиться при різних маніпуляціях з розкладом. Також вона має зовнішній ключ до шаблону тижневого розкладу (тип зв’язку «один до багатьох»).

Рис. 2.4.9. «week»

Розклад

Тепер, коли визначені сутності усіх потрібних нам даних, можна розпочати роботу над самим розкладом, який будуть бачити в додатку користувачі сервісу «Rozcloud», а саме – розкладу на тиждень.

Рис. 2.4.10. «week_template»

Так як тижневий розклад буде здебільшого дублюватись, створимо для нього сутність шаблону на яку буде посилатись сутність тижня. Це дозволить нам не дублювати данні для кожного конкретного тижня, а створити декілька шаблонів тижневого розкладу і потім зв’язувати їх з конкретним навчальним тижнем.

У цієї сутності є звонішній ключ до набору занять з визначеним типом та кількістю годин на тиждень (зв’язок типу «багато до багатьох») для фіксування відповідності створюваного розкладу до графіку освітнього процесу, а також поля днів тижня, які теж є зовнішніми ключами до однієї сутності – шаблону дня (тип зв’язку «один до багатьох»).

Так як потрібно мати можливість маніпулювати конкретним днем, треба створити для нього окрему сутність.

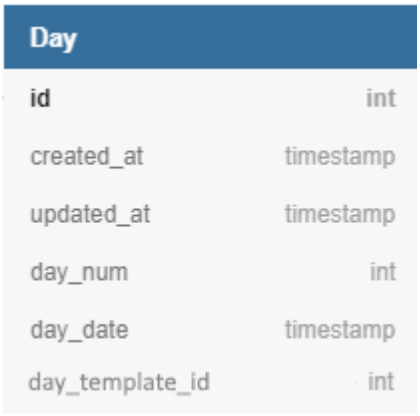

Рис. 2.4.11. «day»

Вона буде мати дату та номер (від 1 до 7) для позначення дня тижня. Проте прив'язувати розклад на день до цієї сутності є недоцільним через велику кількість дублікатів, що виникнуть в такому випадку. Тому треба на зразок сутностей тижневого розкладу створити шаблон дня, на який буде посилатися конкретний день, за допомогою поля зовнішнього ключа (тип зв'язку «один до багатьох»).

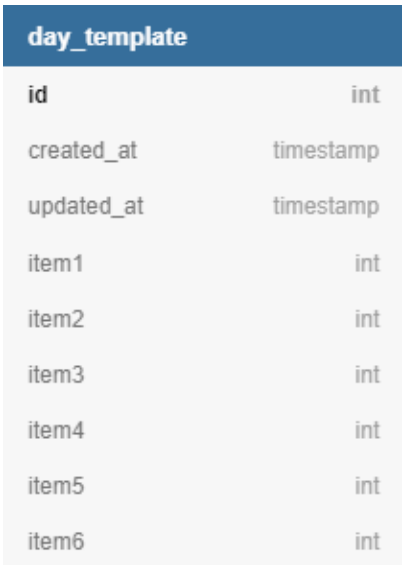

Рис. 2.4.12. «day_template»

В ньому в свою чергу атрибутами будуть зовнішні ключі (тип зв'язку «один до багатьох»), які зв'язують його з сутністю пари. Таких ключів наразі буде 6.

Очевидно що пара буде виступати окремою сутністю. Це дозволить зменшити об’єм бази даних в десятки разів. У сутності пари всі поля окрім первинного ключа та типу пари є зовнішніми з типом зв’язку «один до багатьох»: предмет, викладач, аудиторія та тип пари. Викладач визначається автоматично в залежності від предмету та типу пари по даним у сутності робочого плану. Іншими словами створюючи пару в адмін-панелі, співробітник вибирає лише предмет та аудиторію з випадаючого списку.

Рис. 2.4.13. «lesson»

На цьому роботу над базою даних завершено.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи була поставлена мета розробити базу даних для сервісу «Rozcloud», досягнути максимальної оптимальності у швидкодії бази даних та об’ємах зберігання даних. Для досягнення мети були виконані наступні завдання:

- Визначено вимоги до продукту.
- Проаналізовано аналогічні сервіси.
- Визначено вимоги до БД.
- Визначено основні сутності.
- Визначено атрибути сутностей.
- Визначено зв’язки між сутностями.
- Досліджено сучасні інструменти для розробки бази даних.
- Розроблено базу даних продукту.

Поставлену мету - спроектувати та розробити базу даних для сервісу «Rozcloud» - досягнуто.

Спроектована база даних відповідає всім визначеним вимогам та готова до експлуатації сервісу «Rozcloud». При цьому досягнена максимальна оптимальність у швидкодії та об’ємах зберігання даних.

Також розроблена база даних досить гнучка і готова до змін та подальшого масштабування. Завдяки обраним інструментам розробки – досягнуто лаконічного та якісного захисту передачі даних між клієнтською та серверною частиною.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) Руководство Django [Електронний ресурс] - Режим доступу: https://tutorial.djangogirls.org/ru/how_the_internet_works/

2) Введение в серверную часть [Електронний ресурс] - Режим доступу: https://developer.mozilla.org/ru/docs/Learn/Server-side/First_steps/Introduction

3) Грамотная клиент-серверная архитектура: как правильно проектировать и разрабатывать web API [Електронний ресурс] - Режим доступу: https://tproger.ru/articles/web-api/

4) Документация Django и библиотек для Django [Електронний ресурс] - Режим доступу: https://django.fun/docs/

5) Как спроектировать схему базы данных [Електронний ресурс] - Режим доступу: https://eax.me/database-design/

6) Проектирование баз данных [Електронний ресурс] - Режим доступу: https://spravochnick.ru/bazy_dannyh/proektirovanie_baz_dannyh/

7) Руководство по проектированию реляционных баз данных [Електронний ресурс] - Режим доступу: https://habr.com/ru/post/193136/

8) Основы проектирования баз данных [Електронний ресурс] - Режим доступу: https://sites.google.com/site/anisimovkhv/learning/pris/lecture/tema7/tema7_1

9) Neo4j vs MySQL [Електронний ресурс] - Режим доступу: https://habr.com/ru/post/258179/

10) Основні поняття баз даних [Електронний ресурс] - Режим доступу: http://inf.susu.ac.ru/Klinachev/lc_sga_26.htm

11) Benchmark: PostgreSQL, MongoDB, Neo4j, OrientDB and ArangoDB [Електронний ресурс] - Режим доступу: https://www.arangodb.com/2015/10/benchmark-postgresql-mongodb-arangodb/

12) Пример связи “многие-ко-многим” [Електронний ресурс] - Режим доступу: https://ru.stackoverflow.com/questions/69743/Пример-связи-многие-ко-многим

13) Модели и базы данных [Електронний ресурс] - Режим доступу: https://djbook.ru/rel1.9/topics/db/index.html

14) Создание моделей и миграции базы данных [Електронний ресурс] - Режим доступу: https://metanit.com/python/django/5.1.php

15) Django, начало работы с базой данных [Електронний ресурс] - Режим доступу: https://habr.com/ru/post/87357/

16) Создание сайта на Python/Django: настройка базы данных проекта [Електронний ресурс] - Режим доступу: https://igorosa.com/sozdanie-sajta-na-pythondjango-nastrojka-bazy-dannyx-proekta/

17) Базы данных в Python [Електронний ресурс] - Режим доступу: https://python-scripts.com/database

18) База данных и тестирование [Електронний ресурс] - Режим доступу: https://riptutorial.com/ru/django/example/5356/база-данных-и-тестирование

19) Модели Django [Електронний ресурс] - Режим доступу: https://tutorial.djangogirls.org/ru/django_models/

20) Django+PostgreSQL за 8 шагов [Електронний ресурс] - Режим доступу: https://djbook.ru/examples/77/

21) django.contrib.postgres [Електронний ресурс] - Режим доступу: https://docs.djangoproject.com/en/3.0/ref/contrib/postgres/

22) How To Use PostgreSQL with your Django [Електронний ресурс] - Режим доступу: https://www.digitalocean.com/community/tutorials/how-to-use-postgresql-with-your-django-application-on-ubuntu-14-04

23) How to use PostgreSQL with Django [Електронний ресурс] - Режим доступу: https://www.enterprisedb.com/postgres-tutorials/how-use-postgresql-django

24) Документация к PostgreSQL 9.6.17 [Електронний ресурс] - Режим доступу: https://postgrespro.ru/docs/postgresql/9.6/index

25) How to get started with PostgreSQL [Електронний ресурс] - Режим доступу: https://www.freecodecamp.org/news/how-to-get-started-with-postgresql-9d3bc1dd1b11/

26) Beginner's Guide to PostgreSQL [Електронний ресурс] - Режим доступу: https://www.datacamp.com/community/tutorials/beginners-introduction-postgresql

27) Stored procedures in PostgreSQL [Електронний ресурс] - Режим доступу: https://www.enterprisedb.com/postgres-tutorials/stored-procedures-postgresql-how-create-stored-procedure-and-invoke-it

28) Database Design Standards [Електронний ресурс] - Режим доступу: https://ovid.github.io/articles/database-design-standards.html

29) 11 important database designing rules which I follow [Електронний ресурс] - Режим доступу: https://www.codeproject.com/Articles/359654/11-important-database-designing-rules-which-I-fo-2

30) A Quick-Start Tutorial on Relational Database Design [Електронний ресурс] - Режим доступу: https://www3.ntu.edu.sg/home/ehchua/programming/sql/Relational_Database_Design.html

ДОДАТКИ

Додаток 1

Додаток 1

КОДЕКС АКАДЕМІЧНОЇ ДОБРОЧЕСНОСТІ
ЗДОБУВАЧА ВИЩОЇ ОСВІТИ ХЕРСОНСЬКОГО
ДЕРЖАВНОГО УНІВЕРСИТЕТУ

Я, Заобурний Владислав Вадимович,
учасник(ця) освітнього процесу Херсонського державного університету, **УСВІДОМЛЮЮ**, що академічна доброчесність – це фундаментальна етична цінність усієї академічної спільноти світу.

ЗАЯВЛЯЮ, що у своїй освітній і науковій діяльності **ЗОБОВ'ЯЗУЮСЯ**:

– дотримуватися:

- вимог законодавства України та внутрішніх нормативних документів університету, зокрема Статуту Університету;
- принципів та правил академічної доброчесності;
- нульової толерантності до академічного плагіату;
- моральних норм та правил етичної поведінки;
- толерантного ставлення до інших;
- дотримуватися високого рівня культури спілкування;

– надавати згоду на:

- безпосередню перевірку курсових, кваліфікаційних робіт тощо на ознаки наявності академічного плагіату за допомогою спеціалізованих програмних продуктів;
- оброблення, збереження й розміщення кваліфікаційних робіт у відкритому доступі в інституційному репозитарії;
- використання робіт для перевірки на ознаки наявності академічного плагіату в інших роботах виключно з метою виявлення можливих ознак академічного плагіату;

– самостійно виконувати навчальні завдання, завдання поточного й підсумкового контролю результатів навчання;

– надавати достовірну інформацію щодо результатів власної навчальної (наукової, творчої) діяльності, використаних методик досліджень та джерел інформації;

– не використовувати результати досліджень інших авторів без використання покликань на їхню роботу;

– своєю діяльністю сприяти збереженню та примноженню традицій університету, формуванню його позитивного іміджу;

– не чинити правопорушень і не сприяти їхньому скоєнню іншими особами;

– підтримувати атмосферу довіри, взаємної відповідальності та співпраці в освітньому середовищі;

– поважати честь, гідність та особисту недоторканність особи, незважаючи на її стать, вік, матеріальний стан, соціальне становище, расову належність, релігійні й політичні переконання;

– не дискримінувати людей на підставі академічного статусу, а також за національною, расовою, статевою чи іншою належністю;

– відповідально ставитися до своїх обов'язків, вчасно та сумлінно виконувати необхідні навчальні та науково-дослідницькі завдання;

– запобігати виникненню у своїй діяльності конфлікту інтересів, зокрема не використовувати службових і родинних зв'язків з метою отримання нечесної переваги в навчальній, науковій і трудовій діяльності;

– не брати участі в будь-якій діяльності, пов'язаній із обманом, нечесністю, списуванням, фабрикацією;

– не підроблювати документи;

– не поширювати неправдиву та компрометуючу інформацію про інших здобувачів вищої освіти, викладачів і співробітників;

– не отримувати і не пропонувати винагород за несправедливе отримання будь-яких переваг або здійснення впливу на зміну отриманої академічної оцінки;

– не залякувати й не проявляти агресії та насильства проти інших, сексуальні домагання;

– не завдавати шкоди матеріальним цінностям, матеріально-технічній базі університету та особистій власності інших студентів та/або працівників;

– не використовувати без дозволу ректорату (деканату) символіки університету в заходах, не пов'язаних з діяльністю університету;

– не здійснювати і не заохочувати будь-яких спроб, спрямованих на те, щоб за допомогою нечесних і негідних методів досягати власних корисних цілей;

– не завдавати загрози власному здоров'ю або безпеці іншим студентам та/або працівникам.

УСВІДОМЛЮЮ, що відповідно до чинного законодавства у разі недотримання Кодексу академічної доброчесності буду нести академічну та/або інші види відповідальності й до мене можуть бути застосовані заходи дисциплінарного характеру за порушення принципів академічної доброчесності.

23.04.2020		Владислав Заобурний
(дата)	(підпис)	(ім'я, прізвище)