

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ
Кафедра автоматизованих систем обробки інформації і управління

До захисту допущено:

В.о. завідувача кафедри

_____ *Олександр ПАВЛОВ*
(підпис) (в.ім'я, прізвище)

“ ____ ” _____ 2020 р.

Дипломний проєкт
на здобуття ступеня бакалавра
за освітньо-професійною програмою «Інформаційні управляючі
системи та технології»
спеціальності 122 «Комп'ютерні науки та інформаційні технології»

на тему: «Інформаційна система підтримки діяльності балетної школи»

(комплексна тема)

Загальна частина

Виконали:

студент IV курсу, групи ІС-62

Халін Олег Ігорович

(прізвище, ім'я, по батькові)

(підпис)

студент IV курсу, групи ІС-62

Кравчук Олександр Ігорович

(прізвище, ім'я, по батькові)

(підпис)

студент IV курсу, групи ІС-62

Щеголь Данило Андрійович

(прізвище, ім'я, по батькові)

(підпис)

Керівник

ст. викл. Мажара Ольга Олександрівна

(посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові)

(підпис)

**Консультант з
графічної
документації**

доц., к.т.н., доц. Новінський Валерій Петрович

(посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові)

(підпис)

Рецензент

доц., к.т.н., доц. Михайлова Ірина Юріївна

(посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові)

(підпис)

Засвідчуємо, що у цьому дипломному проєкті
немає запозичень з праць інших авторів без
відповідних посилань.

Студенти _____

(підпис)

(підпис)

Київ – 2020 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет (інститут) інформатики та обчислювальної техніки
(повна назва)

Кафедра автоматизованих систем обробки інформації і управління
(повна назва)

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 122 «Комп'ютерні науки та інформаційні технології»

Освітньо-професійна програма «Інформаційні управляючі системи та технології»

ЗАТВЕРДЖУЮ

В.о. завідувача кафедри

Олександр ПАВЛОВ
(підпис) (в.ім'я, прізвище)

“ ” 2020 р.

ЗАВДАННЯ
на дипломний проєкт студенту

Халіну Олегу Ігоровичу
(прізвище, ім'я, по батькові)

1. Тема проєкту *«Інформаційна система підтримки діяльності
балетної школи» (комплексна тема)*

керівник проєкту Мажара Ольга Олександрівна, ст. викладач

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від “7” травня 2020 р. №1081-с

2. Термін подання студентом проєкту “04” червня 2020 року

3. Вихідні дані до проєкту

Технічне завдання

4. Зміст пояснювальної записки

1. Загальні положення: основні визначення та терміни, опис предметного середовища, огляд ринку програмних продуктів, постановка задачі

2. Інформаційне забезпечення: вхідні дані, вихідні дані, опис структури бази даних

3. Математичне забезпечення: змістовна та математична постановки задачі, обґрунтування та опис методу розв'язання

4. Програмне та технічне забезпечення: засоби розробки, вимоги до технічного забезпечення, архітектура програмного забезпечення, побудова звітів

5. Технологічний розділ: керівництво користувача, методика випробувань програмного продукту

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
-------	--------	------------	--------------	---------------------	----------

5. Перелік графічного матеріалу

1. Схема структурна варіантів використань

2. Схема бази даних

3. Схема структурна класів програмного забезпечення

4. Схема структурна послідовності

5. Схема структурна компонентів

6. Схема структурна розгортання програмного забезпечення

7. Креслення вигляду екранних форм

6. Консультанти розділів проєкту

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

7. Дата видачі завдання «13» квітня 2020 року

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	Вивчення рекомендованої літератури	14.04.2020	
2.	Аналіз існуючих методів розв'язання задачі	15.04.2020	
3.	Постановка та формалізація задачі	16.04.2020	
4.	Розробка інформаційного забезпечення	17.05.2020	
5.	Алгоритмізація задачі	19.04.2020	
6.	Обґрунтування використовуваних технічних засобів	20.04.2020	
7.	Розробка програмного забезпечення	05.05.2020	
8.	Налагодження програми	08.05.2020	
9.	Виконання графічних документів	10.05.2020	
10.	Оформлення пояснювальної записки	13.05.2020	
11.	Подання ДП на попередній захист	15.05.2020	
12.	Подання ДП на основний захист	08.06.2020	
13.	Подання ДП рецензенту	09.06.2020	

Студенти

Олег ХАЛІН
Олександр КРАВЧУК
Данило ЩЕГОЛЬ

Керівник

Ольга МАЖАРА

1	A4		Завдання на дипломний проєкт		2	
2	A4	ДП 6226.00.000 ПЗ	Пояснювальна записка загальна		36	
3	A4	ДП 6226.00.001 ПЗ	Пояснювальна записка індивідуальна		113	
			Халін О. І.			
4	A4	ДП 6211.00.002 ПЗ	Пояснювальна записка індивідуальна		102	
			Кравчук О. І.			
5	A4	ДП 6228.00.003 ПЗ	Пояснювальна записка індивідуальна		62	
			Щеголь Д. А.			
6	A4	ДП 6228.01.000 ТЗ	Технічне завдання		10	
7	A3	ДП 6228.02.000 ССВ	Схема структурна варіантів використання		1	
8	A3	ДП 6228.03.000 СБД	Схема бази даних		1	
9	A3	ДП 6228.04.000 ССК	Схема структурна класів програмного забезпечення		1	
10	A3	ДП 6228.05.000 ССП	Схема структурна послідовності		3	
11	A3	ДП 6228.06.000 ССК	Схема структурна компонентів		1	
12	A3	ДП 6228.07.000 КЕ	Креслення вигляду екранних форм		2	
			ДП 6226 00.000.00			
Зм.	Арк.	ПІБ	Підп.	Дата	Відомість дипломного проєкту	
Розробн.		Халін О.І.				
Керівн.		Мажара О.О.				
Консульт.		Новінський В.П.				
Н/контр.		Новінський В.П.				
В.о.зав.каф.		Павлов О.А.				
					Літ.	Лист
						1
					Листів	1
					КПІ ім. Ігоря Сікорського Каф. АСОІУ Гр. ІС-62	

Пояснювальна записка до дипломного проєкту

на тему: Інформаційна система підтримки діяльності балетної школи.
(комплексна тема)

Київ – 2020 року

АНОТАЦІЯ

Структура та обсяг роботи. Пояснювальна записка дипломного проєкту складається з трьох розділів, містить 4 рисунків, 2 таблиці, 2 додатка, 20 джерел.

Призначенням дипломного проєкту є створення інформаційної системи підтримки діяльності балетної школи. Її реалізація у вигляді веб-застосунку, розміщеного на сервері та мобільного додатку, розміщеного на AppStore[1].

На сьогодні представлення компанії в інтернет просторі є необхідною умовою для її сприйняття користувачами. Найпоширенішою формою присутності у мережі є персоніфікована веб-сторінка. Веб-застосунок слугує не лише платформою для розміщення інформації, але й системою, що допомагає автоматизувати бізнес-процеси. Існує тенденція до створення окремих веб- та моб-додатків поряд з веб-сайтами (наприклад Instagram[2], Facebook[3], Privat24[4]). Актуальною є задача розробки персоніфікованих інформаційних систем для представників малого бізнесу та стартапів.

У розділі інформаційного забезпечення задаються необхідні вимоги до даних, якими оперує система. У нашому випадку це файли бази даних.

При виборі мов програмування один з основних критеріїв була швидкість розробки та простота підтримки. Для серверної частини була вибрана мова Python[5] з простим синтаксисом і великою спільнотою розробників. Для веб застосування використовується TypeScript[6] - мова, що розширює можливості JavaScript[7], завдяки системі типів застосунків має вищу надійність. Нативний мобільний застосунок розроблено на мові Swift[8], яка є новішою і має ширші можливості за Objective-c[9].

					ДП ІС-6226.00.000 ПЗ		
		Прізвище	Підпис	Дата			
Розроб.		Халін О.І.			Інформаційна система підтримки діяльності балетної школи (комплексна тема)	Літ	Арк.
							Аркуші
Керівн.		Мажара О. О.				2	36
Н. кон.		Новінський В.П.				КПІ ім. Ігоря Сікорського Каф. АСОІУ гр. ІС-62	
Затв.		Павлов О.А.					

ДП ІС-6226.00.000 ПЗ

Розділ математичного забезпечення присвячений вирішенню задачі теорії розкладів та обґрунтуванню використаних методів розв'язку. У ньому пояснюється обраний метод та доводиться його доцільність.

БАЛЕТНА ШКОЛА, ІНФОРМАЦІЙНИЙ ДОДАТОК, ВЕБ-ЗАСТОСУНОК, МОБІЛЬНИЙ ДОДАТОК

					ДП ІС-6226.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		3

ABSTRACT

Structure and scope of the research work. There are three chapters, 4 images, 2 tables, 2 supplements, 20 sources in this explanatory note.

The purpose of the graduation project is to create an information system in order to support the activities of the ballet school. Its implementation in a web application form that will be hosted on a server and a mobile application that will be hosted on the App Store[1].

In nowadays' society, the company's representation in the Internet is a necessary condition for its perception by users. The most common form of online presence is a personalized web page. A web application serves not only as a platform for posting information, but also as a system that helps automate business processes. There is a tendency to create separate web and mobile applications along with websites (eg Instagram[2], Facebook[3], Privat24[4]). The task of developing personalized information systems for small businesses and startups is relevant.

The information support section defines the necessary requirements for the data that the system operates. In our case, these are database files.

When choosing programming languages, one of the main criteria was the speed of development and ease of support. Python[5] with a simple syntax and a large community of developers was chosen for the server part. For web applications, TypeScript[6] is used - a language that extends the capabilities of JavaScript[7], thanks to the system of application types has a higher reliability. The native mobile application is developed in the Swift[8] language, which is newer and has more features than Objective-c[9].

The section of mathematical support is devoted to solving the problem of schedule theory and substantiation of the used methods of solution. It explains the chosen method and proves its feasibility.

					ДП IC-6226.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		4

ДП ІС-6226.00.000 ПЗ

BALLET SCHOOL, INFORMATION SYSTEM, WEB APPLICATION,
MOBILE APPLICATION

					ДП ІС-6226.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		5

ЗМІСТ

ВСТУП	8
1 ЗАГАЛЬНІ ПОЛОЖЕННЯ	10
1.1 ОПИС ПРЕДМЕТНОГО СЕРЕДОВИЩА	10
1.1.1 Опис процесу діяльності.....	10
1.1.2 Опис функціональної моделі.....	11
1.2 ОГЛЯД НАЯВНИХ АНАЛОГІВ.....	14
1.2.1 BasePlan	14
1.2.2 Отмечалка.....	15
1.2.3 Mobifitness	15
1.2.4 Парашан	15
1.3 ПОСТАНОВКА ЗАДАЧІ	17
1.3.1 Призначення розробки	17
1.3.2 Цілі та задачі розробки.....	17
Висновок до розділу	19
2 ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ	20
2.1 ВХІДНІ ДАНІ.....	20
2.3 СТРУКТУРА МАСИВІВ ІНФОРМАЦІЇ.....	22
Висновок до розділу	27
3 МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ.....	28
3.1 Змістовна постановка задачі.....	28
3.2 Математична постановка задачі.....	28
3.3 Обґрунтування методу розв'язання.....	30
3.4 Опис методів розв'язання.....	32
Висновок до розділу	34
ЗАГАЛЬНІ ВИСНОВКИ	35
ПЕРЕЛІК ПОСИЛАНЬ.....	36

ДП ІС-6226.00.000 ПЗ

ДП ІС-6226.00.000 ПЗ

Арк.

7

Змн.	Арк.	№ докум.	Підпис	Дата
------	------	----------	--------	------

ВСТУП

У сучасному світі для залучення клієнтів малий бізнес повинен бути представлений у світовій мережі Інтернет. Бізнес повинен бути уніфікованим, персоніфікованим, ефективно використовувати ресурс, щоб виділитись з поміж конкурентів та нести цінність для клієнта. Для ефективної комунікації між бізнесом та його клієнтами, а також для підтримки бізнес — процесів, використовуються кастомізовані CRM/ERP системи. Слід враховувати, що сучасні CRM/ERP системи мають доволі широку функціональність. Освоєння, налаштування, підтримка таких системи потребує багато зусиль, часу, а також спеціальних знань. Тому створення системи, яка відповідає потребам конкретного замовника і містить лише необхідний функціонал є актуальною задачею.

Переважає більшість CRM/ERP орієнтовані на сам бізнес, а не на його клієнтів. В той же час клієнтоорієнтований підхід вимагає створення систем взаємодії бізнесу з користувачами. Особливо чітко таку тенденцію можна прослідкувати для нішового бізнесу, наприклад, балетної школи для дорослих.

Наявність мобільного додатку, який можна використовувати без доступу або при обмеженому доступі до мереж Інтернет — це перевага, яку може запропонувати далеко не кожна система взаємодії з клієнтами. Тому створення не лише веб-версії системи, але й мобільного застосунку підвищує конкурентоспроможність бізнесу та має практичне значення.

Дипломний проект присвячено розробці системи підтримки діяльності балетної школи. Переважає більшість CRM/ERP балетних шкіл є продуктом російського виробництва. Локалізацій таких систем не підтримується виробником, що унеможлиблює їх використання в Україні. Тому в роботі запропоновано спеціалізоване рішення для мережі балетних шкіл, яка працює у місті Києві.

Інформаційна система реалізована у вигляді веб-сайту та мобільного застосунку.

Систему реалізовано для трьох типів користувачів: учня, тренера, адміністратора. Для тренерів та учнів система забезпечує інформаційну підтримку, автоматизує процес складання розкладу, засоби зворотного зв'язку та обслуговування абонементів. Адміністратор має доступ до розширеного функціоналу, зокрема, до статистики відвідуваності занять, навантаження на педагогів, завантаженість учбових залів.

Практичне значення одержаних результатів. Розроблено інформаційну систему підтримки діяльності балетної школи та прийнято її в експлуатацію у мережу балетних шкіл «MaryBallet»[11].

					ДП ІС-6226.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		9

1 ЗАГАЛЬНІ ПОЛОЖЕННЯ

1.1 Опис предметного середовища

1.1.1 Опис процесу діяльності

Ми розробляємо інформаційну систему підтримки процесів діяльності балетної школи, її реалізацію у вигляді веб та мобільного додатку. Однак ця система може використовуватись будь-якою навчальною організацією: починаючи з курсів і персональних занять та закінчуючи великими школами та навіть університетами. Вона є не просто корисною, а необхідною для значної кількості бізнесів, які переростають з маленьких персональних фірм до великих корпорацій, що обслуговують значну кількість людей.

На ринку існує безліч різноманітних CRM/ERP систем, якими користуються більшість розвинутих компаній, однак ми націлились на необхідні користувачу задачі, які не були реалізовані в більшості варіантів. А саме: автоматизація розкладу занять, що включають в себе різні рівні підготовки клієнтів, найбільш зручну для них локацію, час та певного тренера; вибір необхідного для клієнта абонементу, зручний перенос і подовження занять, що може відбуватися без витрачених ресурсів бізнесу; оплата онлайн; можливість тренера відмічати відвідуваність клієнтів і додавати коментарі, що допомагає йому стежити за необхідною навантажкою кожного клієнта індивідуально; автоматизація так званих «розумних повідомлень», що реалізуються за допомогою емейла та смс. Додаємо до цього всього зручний інтерфейс і кросплатформеність, щоб тренер міг це зробити у мобільному додатку в режимі офлайн. В результаті матимемо зручну систему, якою зможуть користуватися більшість шкіл і навчальних організацій.

Ми створюємо продукт персонально за заданими умовами і можемо в будь-який час його кастомізувати під потреби користувача. Для нас це не просто теорія або тестовий проект, для нас це реальна розробка, яка буду

					ДП ІС-6226.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		10

неодмінно в майбутньому приносити користь. Її функціонал буде економити час та гроші власнику, а детально розроблений дизайн і юзер флоу клієнтської частини застосунку дозволить користувачам легко і інтуїтивно зрозуміло виконати всі необхідні задачі та дізнатися всю необхідну інформацію.

1.1.2 Опис функціональної моделі

Програмний продукт буде реалізовано у вигляді веб-застосунку, який буде розміщений на сервері. Користувачі матимуть змогу взаємодіяти з інтерфейсом застосунку за допомогою браузера.

У системі присутні 4 типи акторів:

1. Адміністратор;
2. Анонімний (неавторизований) користувач;
3. Учень;
4. Викладач.]

Примечание [11]: Нумерация

Для кожного користувача в базі даних визначений його тип. Тому після авторизації анонімний користувач змінить свій тип на той, який визначений у базі даних.

У таблиці 1.1 описані актори, варіанти використання та їх описи дій.

					ДП ІС-6226.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		11

Таблиця 1.1 – Актори, варіанти використання та їх описи дій

Актор	Варіант використання	Опис дії варіанта використання
Адміністратор	Додавання/редагування/видалення ресурсів сайту	Адміністратор переходить в пункт ресурсу в адмінці і має можливість проглянути список цих ресурсів. Якщо ресурс підтримує операцію додавання, то на сторінці буде кнопка “Создать”, після натискання якої відкриється форма додавання ресурсу. Якщо ресурс підтримує видалення, то на сторінці редагування ресурсу буде присутня кнопка “Удалить”. Кожен ресурс має свою унікальну форму редагування/додавання, а також набір операцій. Ресурси: – Зали (додавання, редагування, видалення); – Сторінки (редагування); – Шаблони уроку (додавання, редагування, видалення); – Локації (додавання, редагування, видалення); – Користувачі (додавання, редагування, видалення); – Абонемент (додавання, редагування, видалення); – Відвідування (редагування); – Історія оплат (лише перегляд); – Партнери (додавання, редагування, видалення); – Переноси уроків (додавання, редагування, видалення); – Заморозки абонементів (додавання, редагування, видалення).

Примечание [12]: Правила нумерации

Продовження таблиці 1.1

Анонімний користувач	Вхід	Користувач може авторизуватись заповнивши свій номер телефону та пароль, якщо дані будуть вказані вірно користувач стане авторизованим
Анонімний користувач	Реєстрація	Користувач може зареєструватись увівши своє ім'я та номер телефону, після введення користувачу на вказаний номер телефону прийде SMS з паролем для входу
Учень	Покупка абонементу	Учень в своєму особистому кабінеті може обрати абонемент по якому він бажає відвідувати заняття та придбати його.
Учень	Оплата	Учень в особистому кабінеті може оплатити абонемент онлайн. Система відмітить оплату, адміністратори та викладачі зможуть побачити, що користувач успішно оплатив урок.
Учень	Перенесення уроку	Якщо куплений користувачем передбачає перенесення уроків, то користувач має змогу перенести заняття на інший день, для цього йому потрібно обрати день у який проводиться заняття і обрати опцію перенесення
Учень	Заморока абонементу	Якщо куплений користувачем абонемент передбачає перенесення уроків, то користувач має змогу заморозити відвідування на строк, який дозволений абонементом. Система автоматично перенесе всі заняття.
Учень	Перегляд розкладу занять	Учень в своєму особистому кабінеті може переглянути його розклад, визначити коли буде наступний урок.

Продовження таблиці 1.1

Викладач	Відмічання оплати заняття	Викладач може отримати оплату безпосереднього після уроку і має можливість відмітити урок оплаченим для користувача
Викладач	Відмічання відвідування	Викладач може відмітити відвідування користувачем заняття, поставивши відповідну галочку навпроти користувача.
Викладач	Додавання коментарів	Викладач може додати коментар будь-якому учню на занятті, записавши його в необхідний текстовий блок біля користувача.
Викладач	Перегляд розкладу занять	Викладач в своєму особистому кабінеті може переглянути його розклад, визначити коли буде наступний урок.

1.2 Огляд наявних аналогів

Для вирішення задач підтримки функціонування бізнесу, подібного до балетної школи, існують CRM/ERP системи. Однак здебільшого вони містять надлишковий функціонал, який універсальний для різних видів діяльності і не фокусуються на необхідних конкретному виду бізнесу задачах. Крім того, на забезпечення функціоналу щодо організації занять та оплати впливає також географічний регіон. Системи, створені для ринків Америки та Західної Європи складно адаптуються для використання в Україні.

1.2.1 BasePlan

Система розроблена російською компанією. Має в своєму функціоналі базу клієнтів, планувальник і рекламу заходів, модуль постановки задач, контроль за ресурсами компанії, а також емейл і смс повідомлення клієнтам.[12]

Однак даний сервіс не дає можливості використовувати різні способи оплати і можливості визначати різні види абонементів, відмічати присутніх і

жодних функцій, що дають можливість клієнтам взаємодіяти з системою. Також дана система має незручний інтерфейс, що відіграє велику роль на зручності використання.

1.2.2 Отмечалка

Також російське виробництво. Має більш великий функціонал ніж попередні приклади, а саме: робота з усією клієнтською базою, відмічання відвідування, налаштування розкладу, створення абонементів, ведення фінансів.[13]

Дана система доступна тільки у десктопному вигляді, не підтримує емейл і смс розсилки і найголовніше не має жодного представлення для клієнтів. Також не досить зручний і інтуїтивно зрозумілий функціонал.

1.2.3 Mobifitness

Дана система також російського виробника. І має наступний функціонал: облікова система для легкого управління, мобільний додаток для легкої комунікації з клієнтом, онлайн запис на заняття, онлайн оплату.[14]

Даний представник вже має декілька функцій, що дають можливість клієнтам взаємодіяти з системою і емейл/смс повідомлень. Вона має зручніший інтерфейс ніж попередній представник, однак він все ж не є дуже зрозумілим для легкого користування.

1.2.4 Параплан

Найкращий аналог на ринку СНД, однак також є продуктом російського виробництва. Він має досить великий функціонал, доступний і зрозумілий інтерфейс, мобільну версію. В своєму функціоналі сервіс надає наступні можливості: облік клієнтів компанії, додавання і редагування абонементів, планувальник задач для персоналу, аналітика і розрахунок зарплат, емейл і смс повідомлення.[15]

					ДП ІС-6226.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		15

Однак і ця система не має необхідного функціоналу, який був би необхідним для користувачів, які є відвідувачем школи. Не дозволяє тренерам школи контролювати відвідуваність і додавати коментарі щодо кожного клієнта.

Отже, як ми бачимо з аналізованих аналогів тільки один з представлених сервісів пропонує гарне рішення для персонального кабінету студентів школи. Однак сайт системи не надає жодної інформації чи демо версії, яка дозволяє переглянути усі функції і, що також є ключовим фактором, ціну.

Також слід відмітити, що більшість відомих CRM систем, таких як amosm[16], hubspot[17], notion[18], bitrix24[19]... є досить великим продуктами, вони намагаються залучити до свого користування якомога більше користувачів тому додають все більше функціонала. В даному дослідженні як аналоги розглядалися лише вузьконаправленні системи для фітнесу, балету, спортивних секцій тощо.

Усі розглянуті системи розроблені в Росії. Для отримання більш детальної інформації, було проведено опитування служб підтримки, яке виявило відсутність забезпечення платіжних систем України, тому функції прийому платежу для українців недоступні. Також це може означати і те, що при використанні даних сервісів можуть виникати збої, так як більшість російських серверів заборонені на території України, а також, що гарного вітчизняного виробника не існує на ринку.

Особливістю даної роботи є те, що в ній створюється персональний продукт за заданими умовами і передбаченням подальшої переробки та кастомізації. Розробка виконувалась в співпраці з балетною школою та вже впроваджена в використання. Її функціонал дозволяє економити ресурси пам'яті та часу. Детально розроблений дизайн і юзер флоу клієнтської частини застосунку дозволяє користувачам легко і інтуїтивно виконати всі необхідні задачі та дізнатися всю необхідну інформацію.

					ДП ІС-6226.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		16

1.3 Постановка задачі**1.3.1 Призначення розробки**

Розроблювана система призначена для автоматизованої роботи балетної школи. Автоматизація процесів у бізнесі надає купу переваг як підприємцям, так і клієнтам, а саме: значно менші фінансові витрати у довгостроковій перспективі, більша стресостійкість через меншу наявність людського фактору, економія часу та зручніший контроль даних. Окрім того у теперішній час інформаційні системи є невід'ємною частиною корпоративної культури та найціннішою складовою маркетингу.

Інформаційна система повинна бути:

- надійною — для того, щоб усі процеси відбувались без перешкод;
- адаптивною — тобто бути зручною для майбутнього розширення функціоналу або змін у дизайні;
- кросплатформною — для того щоб не обмежувати цільову аудиторію користувачів у рамках певної платформи;
- інтуїтивною у користуванні;
- функціонально повною, але не перенасиченою зайвими функціями.

Даний проект може бути використаний у якості прикладу клієнт-серверного застосунку у будь-якій предметній області. За мету було взято створити як можна гнучкішу архітектуру для подальшого використання у інших кросплатформних систем.

1.3.2 Цілі та задачі розробки

Ціль розробки – створення програмного продукту, який автоматизує створення та зміну індивідуальних розкладів відвідувачів балетної школи,

дозволяє здійснювати електронні платежі для оплати занять, спостерігати за новинами школи та полегшить взаємодію танцюрів та тренерів.

Для досягнення поставленої цілі необхідно виконати наступні задачі:

- розробити дизайн для десктопної та мобільної версій застосунку;
- спроектувати архітектуру серверної частини додатку;
- спроектувати архітектури клієнтських частин додатку для кожної платформи, намагаючись їх якомога більше уніфікувати;
- розробити усі необхідні екрани застосунку згідно дизайну;
- інтегрувати серверну частину у клієнтські частини застосунку;
- розробити платформоорієнтований функціонал, тобто функціонал, який буде унікальним у рамках певної платформи;
- провести тестування додатку на предмет наявності логічних та програмних помилок.

					ДП ІС-6226.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		18

Висновок до розділу

У цьому розділі описано предметне середовище, розглянуто наявні аналоги розробки, описано призначення розробки, обґрунтовано цілі та задачі розробки. Розділ містить відомості про типи користувачів в межах розробленого програмного продукту та функції, які необхідно реалізувати.

					ДП ІС-6226.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		19

2 ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Вхідні дані

Вхідні дані отримуються з наступних джерел:

- Користувачі;
- Адміністратори;
- Платіжні системи.]

Примечание [13]: Правила нумерации

Дані від користувачів надходять з форм в особистих кабінетах користувача, для адмінів це форми в адмініці.

В панелі адміністрування адміністратор може додавати, видаляти та модифікувати наступні дані:

1. Міста;
2. Вулиці;
3. Зали;
4. Користувачі;
5. Уроки;
6. Розклад уроків;
7. Підписки на уроки;
8. Куплені користувачами підписки.]

Примечание [14]: То же самое

Для реєстрації в системі користувач повинен заповнити форму реєстрації, яка включає у себе наступні поля:

1. Ім'я;
2. Номер телефон;
3. Код з смс, яке прийде на номер телефону.]

Проведення платежу при покупці абонементу відбувається на сторонньому сервісі. Таким чином економиться час на розробку, а також знімається відповідальність за безпеку коштів користувачів. Програмний продукт лише отримує підтвердження про здійснення платежу і зараховує покупку підписки.

Вихідні дані

Вихідними даними застосування є інформація, яку користувачі бачать на сторінках:

1. Розклад уроків;
2. Доступні підписки;
3. Найближче заняття;
4. Детальна інформація про заняття;
5. Статистика відвідування заняття.

Приклади наведено у графічних матеріалах у частині звітів екрану.

Примечание [16]:

Примечание [17]:

					ДП ІС-6226.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		21

2.2 Структура масивів інформації

Для розробки застосунку було використано базу даних PostgreSQL[20]

У таблиці 2.1 наведено перелік таблиць, які описують бізнес логіку застосунку.

Таблиця 2.1 – Перелік таблиць, які описують бізнес логіку застосунку.

Таблиця	Опис таблиці	Поля	Тип	Опис поля
city	Місто	name	varchar	Ім'я міста
street	Вулиця	name	varchar	Ім'я вулиці
hall	Зал	id	serial	Ідентифікатор
		city	varchar	Місто у якому знаходиться зал
		street	varchar	Вулиця на якій знаходиться зал
		number	varchar	Номер вулиці
auth_user	Користувач	id	serial	Ідентифікатор
		email	varchar	Емейл
		first_name	varchar	Ім'я
		last_name	varchar	Прізвище
		phone_number	varchar	Номер телефону
		password	binary	Хешований пароль
		avatar_url	varchar	URL на аватар користувача

Продовження таблиці 2.1

		is_coach	boolean	Чи є користувач тренером
		notes	varchar	Додаткові записи
		is_staff	boolean	Чи є користувач персоналом (чи має доступ до панелі адміністрування)
lesson	Урок	id	serial	Ідентифікатор
		lesson_template_id	integer	Шаблон за яким згенерований урок
		hall_id	integer	Зал у якому проводиться заняття
		booking_places_count	integer	Кількість доступних місць на уроці
		trial_places_count	integer	Кількість доступних місць для пробного заняття
		coach_id	integer	Тренер, який проводить заняття
		time	timestamp	Час коли проводитиметься урок
		duration	integer	Тривалість уроку у хвилинах
		skill_level	skilllevel	Рівень підготовки
		type	lesstypetype	Тип уроку

Продовження таблиці 2.1

lesson_template	Шаблон уроку. Те що доступно для запису	id	serial	Ідентифікатор
		hall_id	integer	Зал у якому проводиться заняття
		weekday	integer	День тижня, коли проводиться заняття
		time	time	Час коли проводитиметься урок
		coach_id	integer	Тренер, який проводить заняття
		booking_places_count	integer	Кількість доступних місць на уроці
		trial_places_count	integer	Кількість доступних місць для пробного заняття
		available	boolean	Чи доступний цей шаблон
		duration	integer	Тривалість заняття
		skill_level	skilllevel	Рівень підготовки
		type	lesson_type	Тип уроку

Продовження таблиці 2.1

lesson_visitor	Відвідувач уроку	id	serial	Ідентифікатор
		user_id	integer	Користувач, який відвідує заняття
		lesson_id	integer	Заняття, яке відвідується
		lesson_purchase_id	integer	Куплена підписка, по якій відвідується заняття
		is_first_visit	boolean	Чи це перше відвідування уроку в школі
		pay_amount	integer	Сума, яка повинна бути оплачена за урок (грн)
		moved_to_lesson_id	integer	Урок, на який було перенесене дане відвідування
		visited	boolean	Чи відвідав користувач даний урок
		paid_online	boolean	Оплачено онлайн
		paid_offline	boolean	Оплачено офлайн
lesson_subscription	Підписка на відвідування уроків	id	serial	Ідентифікатор
		lessons_count	integer	Кількість уроків у підписці
		moves_count	integer	Кількість переносів уроків у підписці

Продовження таблиці 2.1

		freeze_days_count	integer	Кількість заморозок уроків у підписці
		price	integer	Ціна підписки
		available	boolean	Чи доступна підписка для покупки
		created_at	timestamp	Час та дата створення підписки
		updated_at	timestamp	Час та дата модифікації підписки
lesson_purchase	Куплена підписка на відвідування уроків.	id	serial	Ідентифікатор
		buyer_id	integer	Користувач, який купив підписку
		lesson_subscription_id	integer	Куплена підписка
		freeze_left	integer	Кількість доступних заморозок
		movements_left	integer	Кількість доступних переносів
		lessons_left	integer	Кількість доступних уроків для відвідування

Висновок до розділу

У розділі інформаційного забезпечення визначаються формати вхідних та вихідних даних та опис структури бази даних. На вхід система отримує дані від різних типів користувачів: учнів, педагогів, адміністраторів. Вихідні дані також формуються в залежності від типу користувача. Для учня це дані про поточний стан абонементу, розклад персональних занять, інформаційні повідомлення тощо. Викладач отримує більш розширену інформацію про наявні групи, заняття, оплату уроків. Адміністратор крім загальних даних має можливість опрацювати інформацію про завантаження залів, впливати на зміни розкладу, отримувати статистику відвідувань для подальшого аналізу. В розділі в окрему таблицю зведено деталізовану структуру даних для всіх типів користувачів.

					ДП ІС-6226.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		27

3 МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Змістовна постановка задачі

У кожного тренера балетної школи є фіксований робочий графік на кожний день його роботи. При проведенні індивідуальних занять у власників балетної школи постає задача: скласти оптимальний розклад для кожного тренера на кожний день роботи зі студентами, що записались на індивідуальні заняття, враховуючи побажання кожного студента щодо часу і тривалості. В балетній школі не один тренер, тому постає задача розробити алгоритм, що за відносно короткий проміжок часу зміг би розробити оптимальний розклад з найменшими втратами часу тренера (вікнами у розкладу).

Розглядається задача теорії розкладів, в якій потрібно знайти оптимальну послідовність індивідуальних занять студентів балетної школи. Кожний студент заповнює форму, в якій вказує наступні дані:

- Тренер, до якого хоче потрапити студент або в якого вже вчиться;
- День, в який необхідно провести заняття;
- Час, в котрий повинен початися урок;
- Тривалість заняття.

3.2 Математична постановка задачі

Потрібно обробити пзаявок студентів на індивідуальне заняття для кожного тренера. Кожна заявка має наступні параметри: фактичний час початку заняття – позначимо як a_i , де $a_i \in [9, 20]$ (балетна школа починає працювати не раніше дев'ятої години ранку та закінчує працювати не пізніше дев'ятої години вечора), та час тривалості заняття – позначимо як l_i , таке що $l_i \in [1, 5]$ (заняття кожного студента не може продовжуватись менше однієї години і більше п'яти годин). Також у кожного тренера є свій графік роботи

					ДП ІС-6226.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		28

на день, позначимо час початку роботи x , таке що $x \in [9, 20]$ та кінець його робочого дня y , таке що $y \in [10, 21], y > x$.

Знайдемо час закінчення заняття студента по формулі 3.1:

$$b_i = a_i + l_i, i = \overline{1, n}, \quad (3.1)$$

Таки чином ми знайшли час закінчення заняття, як суму часу початку заняття і часу тривалості заняття.

Знайдемо час тривалості роботи тренера по формулі 3.2:

$$k = y - x, \quad (3.2)$$

Таким чином ми знайшли тривалість роботи тренера як різницю часу кінця його робочого дня і часу початку роботи.

Тоді цільова функції задачі буде мати наступний вигляд:

$$z = k_i - \sum_{i=0}^n l_i \rightarrow \min, i = \overline{1, n}, \quad (3.3)$$

При виконанні умови 3.4:

$$\begin{cases} a_{i+1} \leq b_i \\ a_0 \geq x \\ b_n \leq y \end{cases}, \quad (3.4)$$

Тобто задача полягає в наступному: визначити порядок в якому повинні бути розташовані заявки на індивідуальні заняття, щоб час простою (коли тренер в робочій час відпочиває, через відсутність занять) був мінімальним.

Графічне представлення умови задачі наведено на рисунку 3.1:

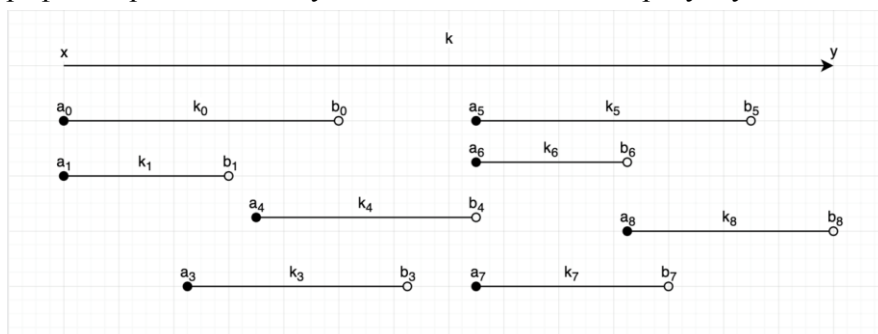


Рисунок 3.1 – Графічне представлення умови задачі

На рисунку 3.2 червоні лінії є часовими проміжками уроків, які входять в оптимальний розклад для тренера. Чорні проміжки при додаванні в розклад зроблять його не оптимальним і утворять більші затримки.

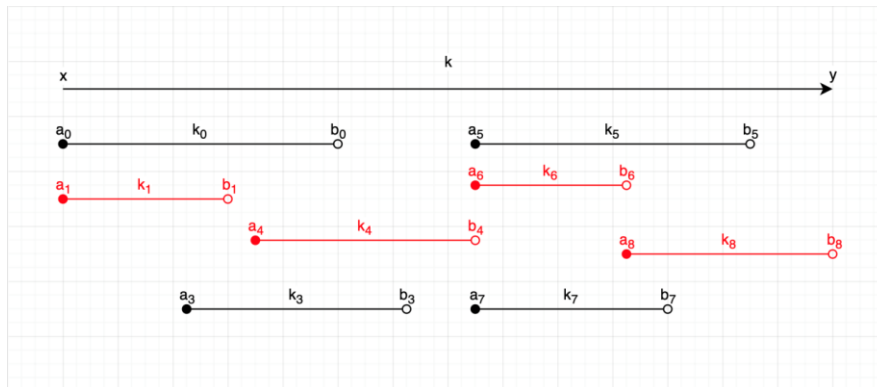


Рисунок 3.2 – Графічне представлення розв'язку задачі

3.3 Обґрунтування методу розв'язання

Часова складність алгоритму

Для вирішення даної задачі запропоновано наступні алгоритми:

- Повний перебір;
- Жадібний алгоритм.]

Примечание [18]:

Повний перебір нам гарантує правильне рішення але часова складність даного алгоритму представлена функцією факторіалу. А значить підходить для пошуку рішення лише для малої кількості учнів. На рисунку 3.3 показаний характер росту функції часу від кількості учнів для повного перебору.

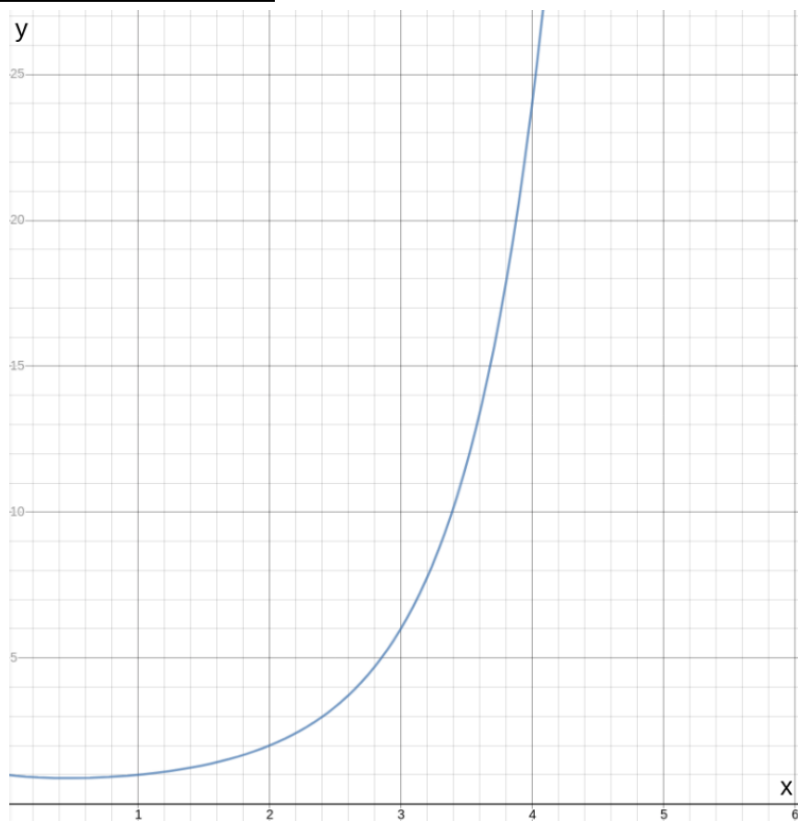


Рисунок 3.3 – Графік характеру зростання функції часу від кількості учнів. По осі абсцис - кількість студентів, по осі ординат - значення функції часу

Якщо кількість студентів буде перевищувати 10, швидкість пошуку оптимального розкладу буде надто низька. Аби прискорити швидкість пошуку вирішено застосувати жадібний алгоритм, який повинен за значно менший час визначити близький до оптимального розклад. Складність жадібного алгоритму: $O(n \cdot \log(n))$

На рисунку 3.4 представлено характер зростання функції часу від кількості студентів для жадібного алгоритму у порівнянні з алгоритмом повного перебору.

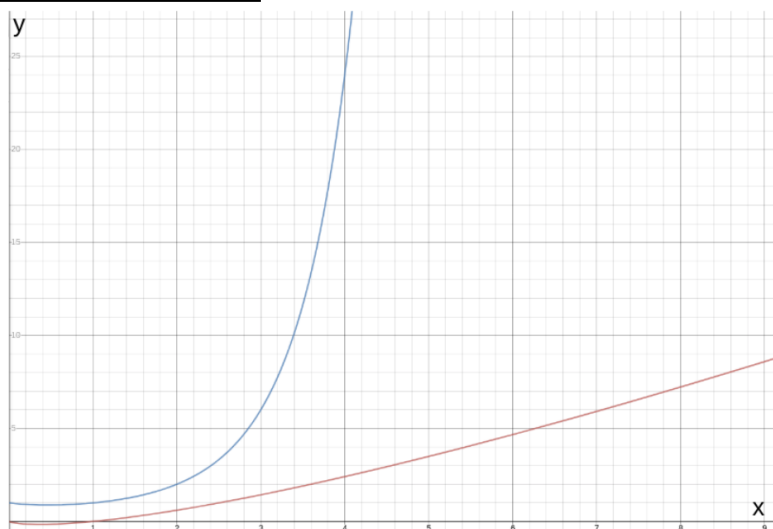


Рисунок 3.4 – Порівняння часу виконання жадібного алгоритму (нижній графік) з алгоритмом повного перебору (верхній графік). По осі абсцис - кількість студентів

Аналіз точності

Точність жадібного алгоритму повністю залежить від характеру даних, чим менша довжина уроку у студентів тим менша теоретична можлива затримка між уроками, а також чим більша кількість студентів, тим більше варіантів для вибору студента з найменшим часом затримки.

При повному переборі, ми гарантовано отримуємо вірний результат.

3.4 Опис методів розв'язання

Наша мета - отримати мінімальний час простою тренера, при розв'язанні жадібним алгоритмом ми намагаємось тримати загальний час простою мінімальним, для цього при виборі студента ми переглядаємо список усіх студентів і обираємо такого студента при якому різниця часу закінчення заняття попереднього студента і часу початку нового студента - мінімальна. Таким чином:

1. На першому кроці обираємо студента у якого різниця часу початку заняття з часом початку роботи тренера - мінімальна

2. Кожен наступний студент обирається так, щоб різниця часу закінчення заняття попереднього студента і обраного була мінімальна серед інших студентів:

$$a_{i+1} - b_i = \min, \text{де}$$

a_{i+1} - час початку заняття нового студенту,

b_i - час закінчення заняття попереднього студенту.

При повному переборі ми перебираємо усі набори студентів і обраховуємо час затримки тренера, при кожному покращенні значення цільової функції ми запам'ятовуємо набір студентів, які дали кращий результат. Після повного перебору усіх варіантів ми отримуємо оптимальне рішення.

Висновок до розділу

У даному розділі запропоновано математичну модель задачі планування розкладу занять студентів у тренера. Було розглянуто два методи пошуку оптимального розкладу та порівняно їх швидкість та точність результатів.

У результаті порівняння кращим з точки зору точності є алгоритм повного перебору, однак при кількості студентів більше 10, час його виконання зростає надто сильно, тому було вирішено використовувати алгоритм повного перебору лише для формування розкладу для тренерів, кількість студентів яких не перевищує 10, таким чином для цих тренерів буде сформований оптимальний розклад за прийнятний час. Якщо ж кількість студентів у тренера перевищує 10, то у цьому випадку застосовується жадібний алгоритм, який дає наближений до оптимального результат за значно менший час.

Таким чином поєднанням точного алгоритму для маленької кількості даних та швидкого неточного алгоритму для великої кількості даних, було досягнуто компроміс між швидкістю та оптимальністю вихідного розкладу.

ЗАГАЛЬНІ ВИСНОВКИ

У дипломному проєкті було розроблено інформаційну систему підтримки діяльності балетної школи у вигляді веб застосунку і мобільного додатку. Розробка системи виконувалась для мережі балетних студій для дорослих «MaryBallet». Важливим аспектом було врахування специфіки бізнесу та забезпечення ефективної комунікації з клієнтом. В той же час, незважаючи на акцент на персоніфікацію продукту, система створювалась з забезпеченням можливої подальшої адаптації для подібних бізнесів, таких як, наприклад, спортивних шкіл, клубів за інтересами або навчальних закладів подібного типу.

Для реалізації клієнтського застосунку було розроблено прототипні версії сайту та протестована їх з участю учнів та педагогів балетної школи. На основі зворотного зв'язку оновлювалися вимоги до інтерфейсу та функціонального наповнення сайту. На основі виокремлених вимог було розроблено дизайн для веб-сайту та мобільної версії застосунку.

В роботі запропоновано та реалізовано загальне архітектурне рішення системи та її складових компонент. Обґрунтовано методи та засоби реалізації системи.

З метою перевірки відповідності вимогам та якості системи було проведено тестування всіх компонент. Крім того, система була надана для користування в тестовому режимі учням та тренерам школи. Для цього було розроблено стратегію розгортання системи в режимі розробки, тестовому та для загального користування.

На сьогодні веб-сайт з базовим функціоналом введено в експлуатацію мережею «MaryBallet», користування мобільним застосунком продовжується в тестовому режимі.

					ДП ІС-6226.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		35

ПЕРЕЛІК ПОСИЛАНЬ

1. AppStore [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу:<https://apps.apple.com/ua/app/apple-store/id375380948?l=ru>;
2. Instagram[Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу:<https://www.instagram.com> ;
3. Facebook[Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу:<https://www.facebook.com> ;
4. Privat24 [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу:<https://www.privat24.ua> ;
5. Python[Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу:<https://www.python.org> ;
6. TypeScript[Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу:<http://typescripqlang.org> ;
7. JavaScript[Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу:<https://javascript.info> ;
8. Swift[Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу:<https://www.apple.com/ru/swift/> ;
9. Objective-c[Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу:<https://developer.apple.com/library/archive/documentation/Cocoa/Conceptual/ProgrammingWithObjectiveC/Introduction/Introduction.html> ;
10. Excel[Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу:<https://www.microsoft.com/uk-ua/microsoft-365/excel> ;
11. Балетная школа в Киеве - студия балета MaryBallet, частные уроки, занятия у станка [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу: <https://maryballet.com.ua/>;
12. Управление клиентами от Baseplan. Crm системы для управления взаимоотношениями с клиентами. [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу: <https://baseplan.ru/>;

13. CRM система автоматизации учета посещаемости занятий, программа для электронного учета клиентов | Отмечалка [Электронный ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу: <https://otmechalka.com/>;

14. Программное обеспечение для управления фитнес-клубом - Mobifitness [Электронный ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу: <https://mobifitness.ru/>;

15. Параплан CRM - облачная CRM [Электронный ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу: <https://paraplancrm.ru/>;

16. Онлайн CRM система. Управление клиентами в современной облачной CRM — amoCRM [Электронный ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу: <https://www.amocrm.ru/>;

17. HubSpot | InboundMarketing, Sales, andServiceSoftware [Электронный ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу: <https://www.hubspot.com/>;

18. Notion – The all-in-one workspace for your notes, tasks, wikis, and databases. [Электронный ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу: <https://www.notion.so/>;

19. Бітрікс24: сервіс автоматизації і оптимізації бізнес-процесів компанії [Электронный ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу: <https://www.bitrix24.ua/>;

20. PostgreSQL: The world's most advanced opensource database [Электронный ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу: <https://www.postgresql.org/>;

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”
Кафедра автоматизованих систем обробки інформації і управління

УЗГОДЖЕНО

Керівник проекту

_____ О. О. Мажара
(підпис) (ініціали, прізвище)

“13” квітня 2020 р.

ЗАТВЕРДЖУЮ

В.о. завідувача кафедри

_____ О.А. Павлов
(підпис) (ініціали, прізвище)

“14” квітня 2020 р.

«Інформаційна система підтримки процесів
діяльності балетної школи»

ТЕХНІЧНЕ ЗАВДАННЯ

Шифр ДП ІС-6226.00.000 ТЗ

на 11 сторінках

Київ – 2020 року

ЗМІСТ

1	ЗАГАЛЬНІ ПОЛОЖЕННЯ	3
1.1	Повне найменування системи та її умовне позначення	3
1.2	Найменування організації-замовника та організації-учасників робіт	3
1.3	Перелік документів, на підставі яких створюється система.....	3
1.4	Планові терміни початку і закінчення роботи зі створення системи.....	3
2	ПРИЗНАЧЕННЯ І ЦІЛІ СТВОРЕННЯ ЗАСОБІВ.....	5
2.1	Призначення засобів.....	5
2.2	Цілі створення засобів	5
3	ХАРАКТЕРИСТИКА ОБ'ЄКТА АВТОМАТИЗАЦІЇ.....	7
4	ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.....	8
4.1	Вимоги до функціональних характеристик.....	8
4.2	Вимоги до функціональних характеристик.....	9
4.3	Вимоги до складу і параметрів технічних засобів.....	9
5	СТАДІЇ І ЕТАПИ РОЗРОБКИ	11
6	ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ	12

					ДП ІС-6226.00.000 ТЗ			
		Прізвище	Підпис	Дата				
Розроб.		Халін О.І.			Інформаційна система підтримки діяльності балетної школи (комплексна тема)	Літ	Арк.	Аркушів
							2	11
Керівн.		Мажара О. О.				КПІ ім. Ігоря Сікорського Каф. АСОІУ гр. ІС-62		
Н. кон.		Новінський В.П.						
Затв.		Павлов О.А.						

1 ЗАГАЛЬНІ ПОЛОЖЕННЯ

1.1 Повне найменування системи та її умовне позначення

Повне найменування системи: «Інформаційна система підтримки діяльності балетної школи».

Коротке найменування системи: «MaryBallet».

1.2 Найменування організації-замовника та організацій-учасників робіт

Замовником проекту є кафедра автоматизованих систем обробки інформації та управління факультету інформатики та обчислювальної техніки Національного технічного університету України «КПІ імені Ігоря Сікорського».

Розробники застосунку – студенти групи ІС-62 факультету інформатики та обчислювальної техніки НТУУ «КПІ імені Ігоря Сікорського» Халін Олег Ігорович, Кравчук Олександр Ігорович, Щеголь Данило Андрійович.

1.3 Перелік документів, на підставі яких створюється система

Підставою для розробки “Інформаційна система підтримки діяльності балетної школи” є завдання на дипломне проектування, затверджене кафедрою автоматизованих систем обробки інформації та управління Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

1.4 Планові терміни початку і закінчення роботи зі створення системи

Плановий термін початку роботи над створенням застосунку – 1 грудня 2019 року.

					ДП ІС-6226.00.000ТЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		3

ДП ІС-6226.00.000ТЗ

Плановий термін по закінченню роботи над створенням застосунку – 25 травня 2020 року.

					ДП ІС-6226.00.000ТЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		4

2 ПРИЗНАЧЕННЯ І ЦІЛІ СТВОРЕННЯ ЗАСОБІВ

2.1 Призначення засобів

Розроблювана система призначена для автоматизованої роботи балетної школи. Автоматизація процесів у бізнесі надає купу переваг як підприємцям, так і клієнтам. Автоматизовані системи мають наступні переваги: значно менші фінансові витрати у довгостроковій перспективі, більша стресостійкість через меншу наявність людського фактору, економія часу та зручніший контроль даних. Окрім того у теперішній час інформаційні системи є невід'ємною частиною корпоративної культури та найціннішою складовою маркетингу.

Інформаційна система повинна бути:

- надійною — для того, щоб усі процеси відбувались без перешкод;
- адаптивною — тобто бути зручною для майбутнього розширення функціоналу або змін у дизайні;
- кросплатформною — для того щоб не обмежувати цільову аудиторію користувачів у рамках певної платформи;
- інтуїтивною у користуванні;
- функціонально повною, але не перенасиченою зайвими функціями.

Даний проект може бути використаний у якості прикладу клієнт-серверного застосунку у будь-якій предметній області. За мету було взято створити як можна гнучкішу архітектуру для подальшого використання у інших кросплатформних систем.

2.2 Цілі створення засобів

Ціль розробки – створення програмного продукту, який автоматизує створення та зміну індивідуальних розкладів відвідувачів балетної школи,

					ДП ІС-6226.00.000ТЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		5

ДП ІС-6226.00.000ТЗ

дозволить здійснювати електронні платежі для оплати занять, спостерігати за новинами школи та полегшить взаємодію танцюрів та тренерів.

Для досягнення поставленої цілі необхідно виконати наступні задачі:

- розробити дизайн для десктопної та мобільної версій застосунку;
- спроектувати архітектуру серверної частини додатку;
- спроектувати архітектури клієнтських частин додатку для кожної платформи, намагаючись їх якомога більше уніфікувати;
- розробка усіх необхідних екранів застосунку згідно дизайну;
- інтеграція серверної частини у клієнтські частини застосунку;
- розробка платформоорієнтованого функціоналу, функціоналу який буде унікальним у рамках певної платформи;
- рефакторинг коду з метою оптимізації його швидкодії та лаконічності;
- тестування додатку на предмет наявності логічних та програмних помилок.

					ДП ІС-6226.00.000ТЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		6

3 ХАРАКТЕРИСТИКА ОБ'ЄКТА АВТОМАТИЗАЦІЇ

В роботі розробляється інформаційна система підтримки процесів діяльності балетної школи у вигляді веб та мобільного додатку. Однак ця система може використовуватись будь-якою навчальною організацією: починаючи з курсів і персональних занять та закінчуючи великими школами та навіть університетами. Вона є не просто корисною, а необхідною для значної кількості бізнесів, які переростають з маленьких персональних фірм до великих корпорацій, що обслуговують значну кількість людей.

На ринку існує безліч різноманітних CRM/ERP систем, якими користуються більшість розвинутих компаній, однак ми націлились на необхідні користувачу задачі, які не були реалізовані в більшості варіантів. А саме: автоматизація розкладу занять, що включають в себе різні рівні підготовки клієнтів, найбільш зручну для них локацію, час та певного тренера; вибір необхідного для клієнта абонементу, зручний перенос і подовження занять, що може відбуватися без витрачених ресурсів бізнесу; оплата онлайн; можливість тренера відмічати відвідуваність клієнтів і додавати коментарі, що допомагає йому стежити за необхідною навантажкою кожного клієнта індивідуально; автоматизація так званих «розумних повідомлень», що реалізуються за допомогою емейла та смс. Додаємо до цього всього зручний інтерфейс і кросплатформеність, щоб тренер міг це зробити у мобільному додатку в режимі офлайн. В результаті матимемо зручну систему, якою зможуть користуватися більшість шкіл і навчальних організацій.

В роботі створюється продукт персонально за заданими, що дозволяє в будь-який час його кастомізувати під потреби користувача. Його функціонал буде економити час та гроші власнику, а детально розроблений дизайн і юзер флоу клієнтської частини застосунку дозволить користувачам легко і інтуїтивно зрозуміло виконати всі необхідні задачі та дізнатися всю необхідну інформацію.

					ДП ІС-6226.00.000ТЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		7

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

Інформаційна система повинна надавати можливість переглядати обраний розклад занять, редагувати розклад, обираючи час та викладачів, а також отримувати повідомлення щодо змін у розкладі.

Система повинна мати бокове меню для зручної навігації по застосунку, яке повинне мати такі вкладки: розклад, повідомлення, поточне заняття, підтримка.

Система повинна виконувати наступні функції:

- Переглянути розклад – користувач обирає вкладку «Розклад» та на екрані відображається розклад занять на тиждень;
- Редагувати розклад – користувач обирає вкладку «Розклад», натискає на кнопку «Редагувати» та отримує можливість видаляти заняття з поточного розкладу та додавати заняття зі списку усіх занять;
- Переглянути інформацію про поточне заняття – користувач системи натискає на вкладку «Поточне заняття» для огляду учасників та інформації про тренера;
- Переглянути статус оплати зайняття – тренер натискає на вкладку «Поточне заняття» та переглядає статус оплати зайняття для кожного відвідувача поточного зайняття;
- Переглянути історію повідомлень – користувач натискає на вкладку «Повідомлення» та переглядає історію отриманих раніше повідомлень;
- Надавати доступ до контактних даних служби підтримки – користувач обирає вкладку «Підтримка» та отримує можливість подзвонити або написати у службу підтримки;

					ДП ІС-6226.00.000ТЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		8

- Система повинна мати змогу надсилати повідомлення окремим користувачам із текстовими даними. Користувачі в свою чергу повинні мати змогу отримувати дані повідомлення.]

Примечание [11]: Правила нумерации

4.2 Вимоги до функціональних характеристик

Система повинна працювати безвідмовно, незважаючи на наявність ймовірних дефектів, котрі можуть проявлятися під час експлуатації. Жодні зовнішні чи внутрішні умови не повинні бути причиною руйнування даних в інформаційних сховищах, збою чи закінченню життєвого циклу програми.

Розрив або відсутність Інтернет-з'єднання не повинні бути причиною критичних помилок у роботі мобільної версії додатку, адже вона підтримує кешування даних та не залежить від Інтернет-з'єднання для відображення певних даних.

Система повинна проходити тестування критичними ситуаціями ще на етапі розробки задля усунення подальших можливих дефектів вчасно.

4.3 Вимоги до складу і параметрів технічних засобів

Для коректного функціонування програмного продукту на стороні користувача повинні бути задовільнені певні вимоги щодо складу і параметрів технічних засобів.

Для мобільного додатку єдиною вимогою буде те, що пристрій користувача повинен працювати на операційній системі iOS 13. Дана вимога пов'язана із використанням найновіших фреймворків під час написання програмного продукту.

Для веб-застосунку список вимог є трохи ширшим, втім даним вимогам задовольняє майже будь-який комп'ютер на сьогоднішній день:

- Процесор з тактовою частотою не менше 1 ГГц;
- Об'єм оперативної пам'яті не менше 1 Гб;
- Підключення до мережі Інтернет.

					ДП ІС-6226.00.000ТЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		9

ДП ІС-6226.00.000ТЗ

Для взаємодією із веб-застосунком комп'ютер також повинен мати необхідну периферію:

- Монітор;
- Миша, трекпад (або дисплей із тачскріном);
- Клавіатура (або дисплей з тачскріном).

					<i>ДП ІС-6226.00.000ТЗ</i>	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		10

5 СТАДІЇ І ЕТАПИ РОЗРОБКИ

Основні етапи виконання робіт з розробки наведені в таблиці 5.1.

Таблиця 5.1

	Назва етапу	Термін виконання
	Вивчення рекомендованої літератури	01.03.2020
	Аналіз існуючих методів розв'язання задачі	07.03.2020
	Постановка та формалізація задачі	15.03.2020
	Розробка програмного забезпечення	11.05.2020
	Налагодження програми	12.05.2020
	Оформлення пояснювальної записки	14.05.2020
	Подання ДП на попередній захист	15.05.2020
	Подання ДП на основний захист	25.06.2020

6 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

Задля перевірки правильності роботи програмного продукту буде проведено функціональне тестування. В ході тестування буде виконано перевірку всіх функціональних характеристик, як веб-застосунку, так і мобільного застосунку. Також, система буде перевірена на відмовостійкість шляхом виконання некоректних дій користувачем та надсиланням запитів із некоректними даними. Окремими тестами буде перевірена безпека системи в цілому. За допомогою спеціальних інструментів будуть перевірені швидкодія та енергоефективність мобільного застосунку.

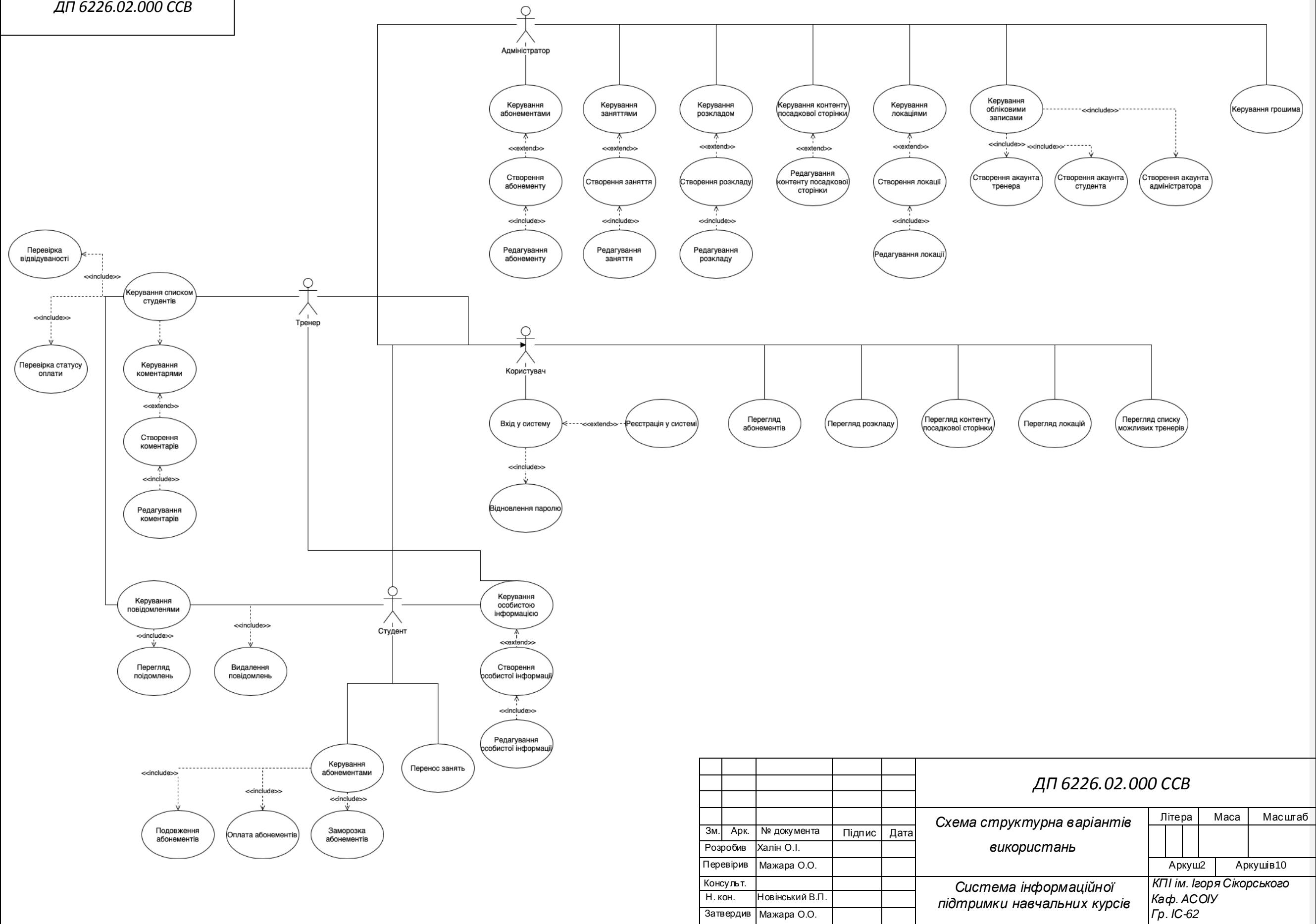
					ДП ІС-6226.00.000ТЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		12

Графічний матеріал до дипломного проєкту

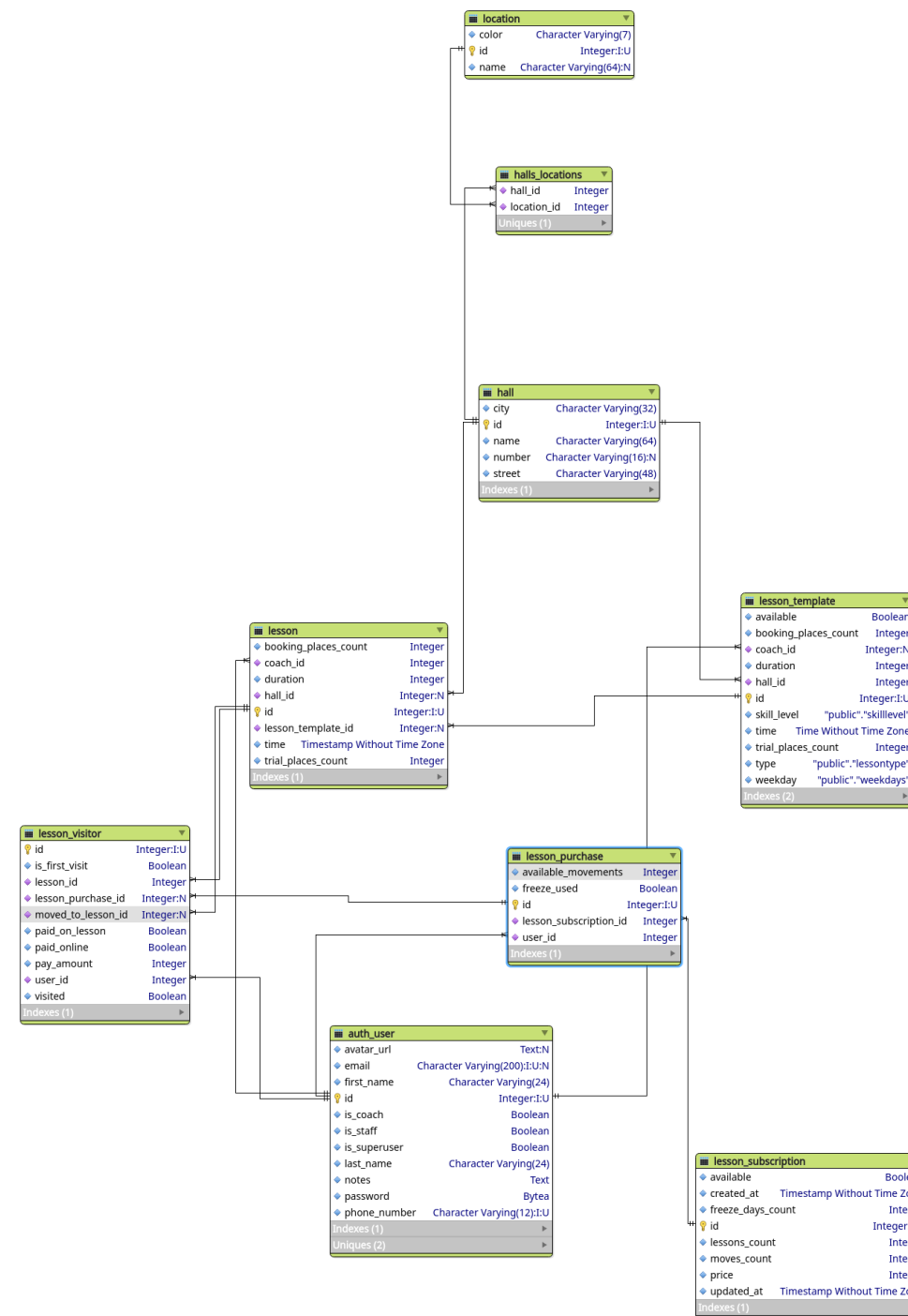
на тему: Інформаційна система підтримки діяльності балетної школи

(комплексна тема)

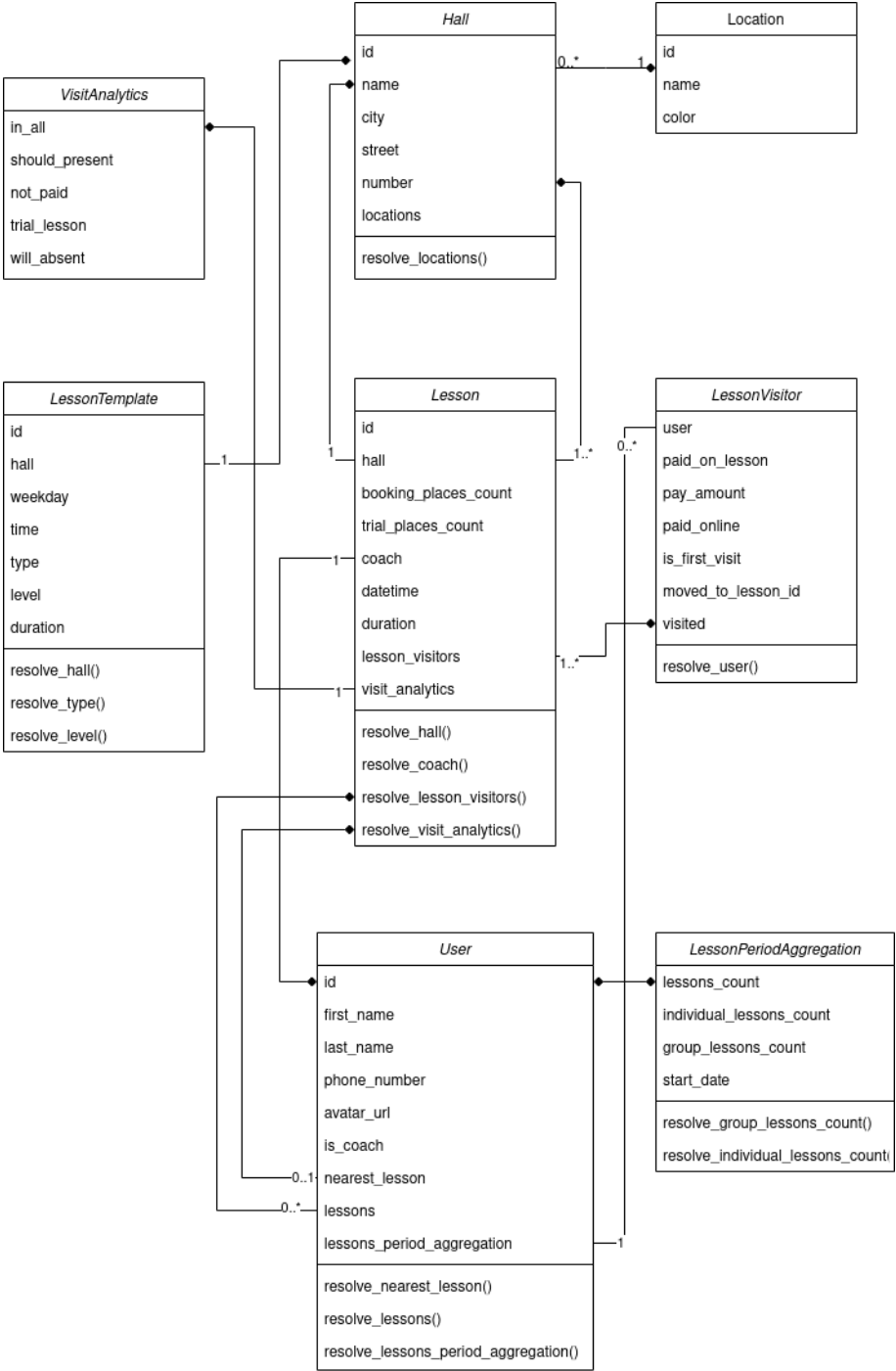
Київ – 2020 року



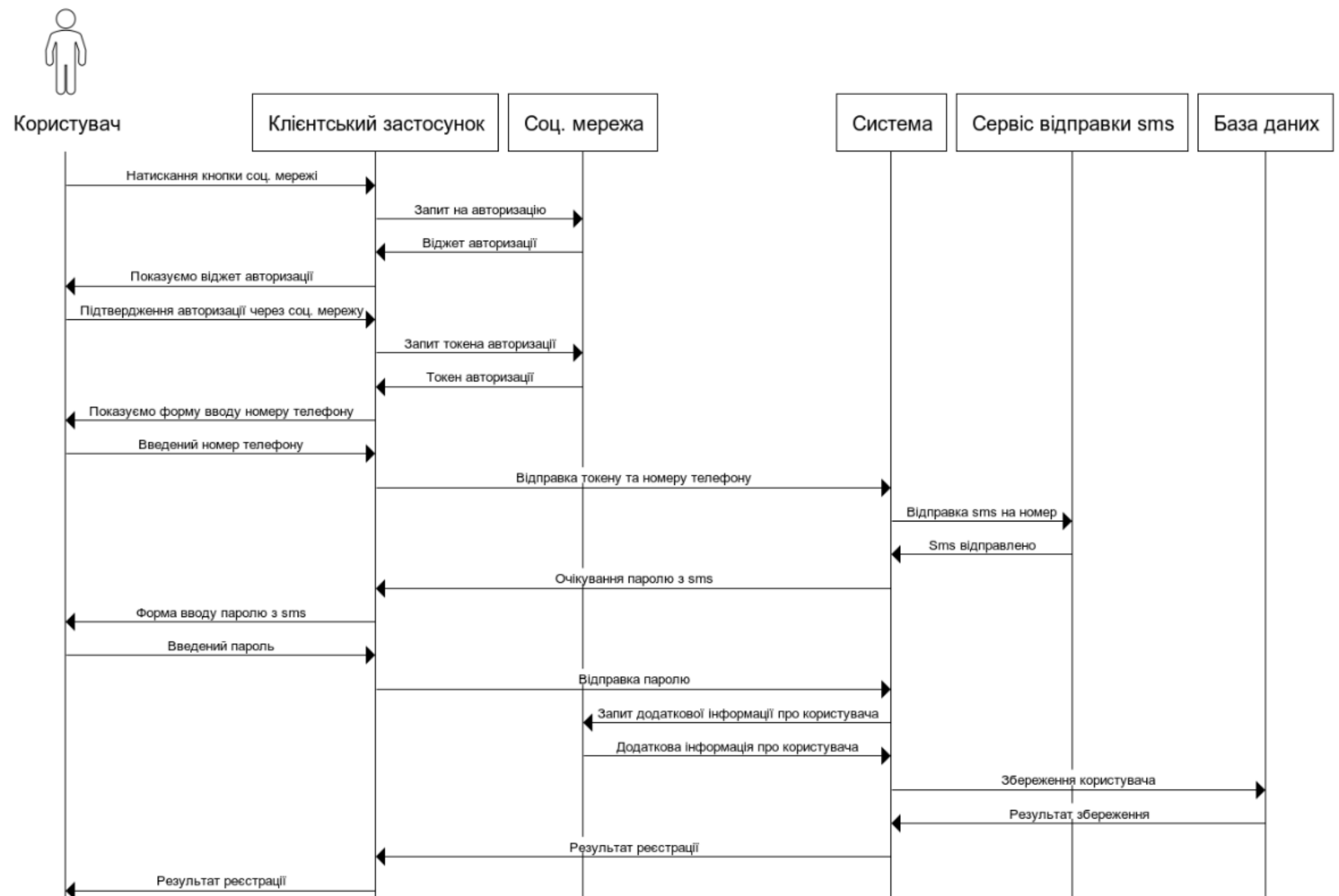
						ДП 6226.02.000 ССВ					
						Схема структурна варіантів використань	Літера		Маса	Масштаб	
							Аркуш2		Аркушів10		
Зм.	Арк.	№ документа	Підпис	Дата		Система інформаційної підтримки навчальних курсів	КПІ ім. Ігоря Сікорського Каф. АСОІУ Гр. ІС-62				
Розробив	Халін О.І.										
Перевірів	Мажара О.О.										
Консульт.											
Н. кон.	Новінський В.П.										
Затвердив	Мажара О.О.										



					ДП 6226.03.000 СБД				
					Схема базиданих	Літера		Маса	Масштаб
Зм.	Арк.	№ документа	Підпис	Дата					
Розробив	Халін О.І.								
Перевірив	Мажара О.О.								
Консульт.						Аркуш 3		Аркушів10	
Н. кон.	Новінський В.П.				Система інформаційної підтримки навчальних курсів	КПІ ім. Ігоря Сікорського Каф. АСОУ Гр. ІС-62			
Затвердив	Мажара О.О.								



						ДП 6226.04.000 ССК			
						Схема структурна класів програмного забезпечення	Літера	Маса	Масштаб
Зм.	Арк.	№ документа	Підпис	Дата					
Розробив	Халін О.І.								
Перевірив	Мажара О.О.					Система інформаційної підтримки навчальних курсів	Аркуш 4		Аркушів10
Консульт.							КПІ ім. Ігоря Сікорського Каф. АСОІУ Гр. ІС-62		
Н. кон.	Новінський В.П.								
Затвердив	Мажара О.О.								

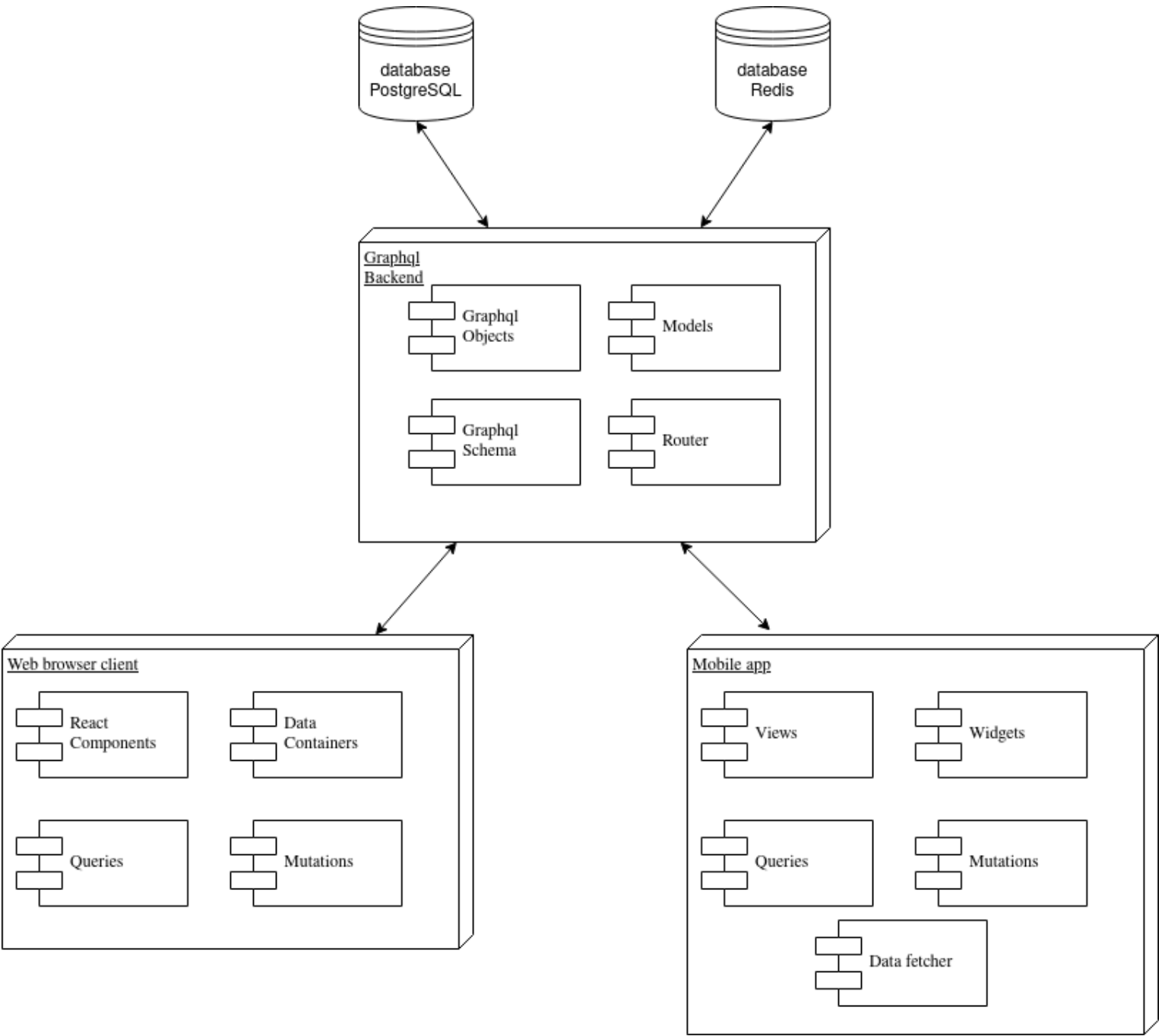


					ДП 6226.05.000 ССП						
					Схема структурна послідовності	Літера		Маса		Масштаб	
Зм.	Арк.	№ документа	Підпис	Дата							
Розробив	Халін О.І.										
Перевірів	Мажара О.О.					Аркуш 5		Аркушів10			
Консульт.					Система інформаційної підтримки навчальних курсів	КПІ ім. Ігоря Сікорського Каф. АСОІУ Гр. ІС-62					
Н. кон.	Новінський В.П.										
Затвердив	Мажара О.О.										

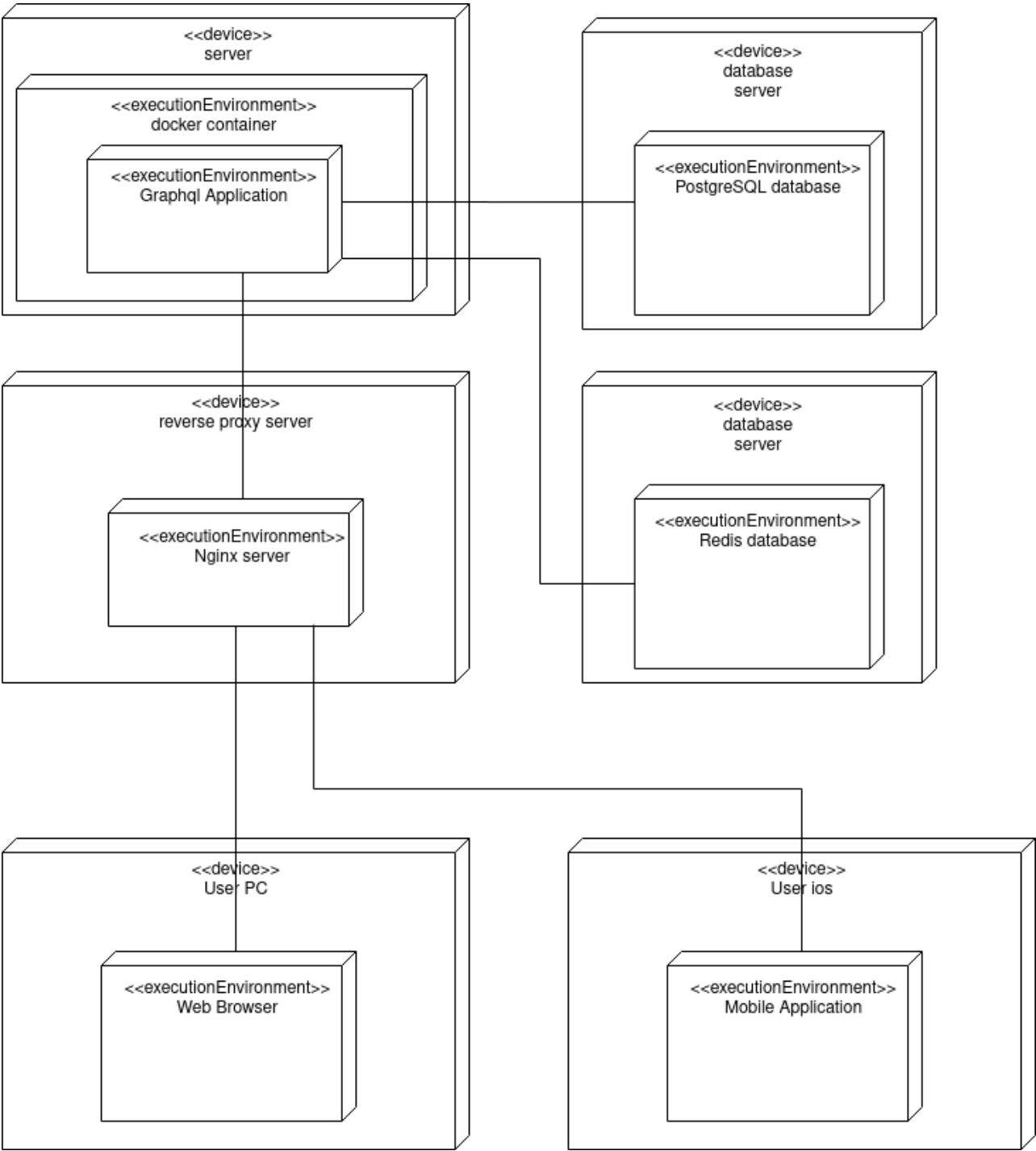
Авторизація



					ДП 6226.05.000 ССП						
					Схема структурна послідовності	Літера		Маса		Масштаб	
Зм.	Арк.	№ документа	Підпис	Дата	Система інформаційної підтримки навчальних курсів	Аркуш 6				Аркушів10	
Розробив	Халін О.І.										
Перевірів	Мажара О.О.										
Консульт.											
Н. кон.	Новінський В.П.				КПІ ім. Ігоря Сікорського Каф. АСОІУ Гр. ІС-62						
Затвердив	Мажара О.О.										



					ДП 6226.06.000 ССК						
					Схема структурних компонентів програмного забезпечення	Літера		Маса		Масштаб	
Зм.	Арк.	№ документа	Підпис	Дата							
Розробив	Халін О.І.										
Перевірив	Мажара О.О.					Аркуш 7		Аркушів 10			
Консульт.											
Н. кон.	Новінський В.П.				Система інформаційної підтримки навчальних курсів	КПІ ім. Ігоря Сікорського Каф. АСОІУ Гр. ІС-62					
Затвердив	Мажара О.О.										



					ДП 6226.06.000 ССК						
					Схема структурна розгортання програмного забезпечення	Літера		Маса		Масштаб	
						Аркуш 8				Аркушів 10	
Зм.	Арк.	№ документа	Підпис	Дата	Система інформаційної підтримки навчальних курсів	КПІ ім. Ігоря Сікорського Каф. АСОІУ Гр. ІС-62					
Розробив	Халін О.І.										
Перевірив	Мажара О.О.										
Консульт.											
Н. кон.	Новінський В.П.										
Затвердив	Мажара О.О.										

ДП 6226.09.000 ЕК

МФВ

Привет, Дмитрий

День

Неделя

Месяц

Расписание

Текущее занятие

Уведомления

Настройки

Вторник 14 июня

25 занятий

Следующее занятие

12:00

ул. Иванова, д. 7

2 не оплачено

2 пробное занятие

2 не придут

11 из 20

14:00

ул. Иванова, д. 7

2 не оплачено

2 пробное занятие

2 не придут

11 из 20

16:00

ул. Иванова, д. 7

2 не оплачено

2 пробное занятие

2 не придут

11 из 20

17:00

ул. Иванова, д. 7

2 не оплачено

2 пробное занятие

2 не придут

11 из 20

18:00

ул. Иванова, д. 7

2 не оплачено

2 пробное занятие

2 не придут

11 из 20

Среда 14 июня

25 занятий

Следующее занятие

12:00

ул. Иванова, д. 7

2 не оплачено

2 пробное занятие

2 не придут

11 из 20

12:00

ул. Иванова, д. 7

2 не оплачено

2 пробное занятие

2 не придут

11 из 20

МФВ

Привет, Дмитрий

День

Неделя

Месяц

Расписание

Текущее занятие

Уведомления

Настройки

13.06 - 19.06

25 занятий

Следующее занятие

13.06 Понедельник

7 занятий

групповые: 3

индивидуальные: 3

Дата: Вторник 14 июня

Время: 12:00

Адрес: ул.Иванова, д.7

Длительность: 1 час 30 минут

14.06 Вторник

7 занятий

групповые: 3

индивидуальные: 3

15.06 Среда

7 занятий

групповые: 3

индивидуальные: 3

16.06 Четверг

7 занятий

групповые: 3

индивидуальные: 3

17.06 Пятница

7 занятий

групповые: 3

индивидуальные: 3

18.06 Суббота

7 занятий

групповые: 3

индивидуальные: 3

20.06 - 26.06

25 занятий

Следующее занятие

20.06 Понедельник

7 занятий

групповые: 3

индивидуальные: 3

21.06 Вторник

7 занятий

групповые: 3

индивидуальные: 3

МФВ

Привет, Дмитрий

День

Неделя

Месяц

Расписание

Текущее занятие

Уведомления

Настройки

06 июня

25 занятий

Следующее занятие

01.06 - 07.06

7 занятий

групповые: 3

индивидуальные: 3

Дата: Вторник 14 июня

Время: 12:00

Адрес: ул.Иванова, д.7

Длительность: 1 час 30 минут

08.06 - 14.06

7 занятий

групповые: 3

индивидуальные: 3

15.06 - 21.06

7 занятий

групповые: 3

индивидуальные: 3

22.06 - 28.06

7 занятий

групповые: 3

индивидуальные: 3

29.06 - 31.06

7 занятий

групповые: 3

индивидуальные: 3

07 июля

25 занятий

Следующее занятие

01.07 - 07.07

7 занятий

групповые: 3

индивидуальные: 3

08.07 - 14.07

7 занятий

групповые: 3

индивидуальные: 3

15.07 - 21.07

7 занятий

групповые: 3

индивидуальные: 3

МФВ

Привет, Дмитрий

День

Неделя

Месяц

Расписание

Текущее занятие

Уведомления

Настройки

Вторник 14 июня

25 занятий

Следующее занятие

Дата: Вторник 14 июня

Время: 12:00

Адрес: ул.Иванова, д.7

Длительность: 1 час 30 минут

2 не оплачено

2 пробное занятие

2 не придут

11 из 20

Ученики

11/20 занятий

Следующее занятие

Максим Иванов

не оплачено 100₽

Добавить комментарий

Иван Кузнецов

не оплачено 130₽

На прошлом занятии вывели купюру, в погашено, все учтено - занятием имеет, но без твоей помощи.

Максим Иванов

оплачено

Максим Иванов

оплачено

Максим Иванов

оплачено

МФВ

Привет, Дмитрий

День

Неделя

Месяц

Расписание

Текущее занятие

Уведомления

Настройки

Вторник 14 июня

25 занятий

Следующее занятие

Дата: Вторник 14 июня

Время: 12:00

Адрес: ул.Иванова, д.7

Длительность: 1 час 30 минут

2 не оплачено

2 пробное занятие

2 не придут

11/20 занятий

Ученики

11/20 занятий

Следующее занятие

Максим Иванов

не оплачено 100₽

Добавить комментарий

Максим Иванов

не оплачено 100₽

Добавить комментарий

Максим Иванов

оплачено

Максим Иванов

оплачено

Максим Иванов

оплачено

МФВ

Привет, Дмитрий

День

Неделя

Месяц

Расписание

Текущее занятие

Уведомления

Настройки

Вторник 14 июня

25 занятий

Следующее занятие

12:00

ул. Иванова, д. 7

2 не оплачено

2 пробное занятие

2 не придут

11 из 20

Ученик Дмитрий Швец

сменил номер телефона, имя и фамилию

14.06

2 не оплачено

2 пробное занятие

2 не придут

11 из 20

Ученик Дмитрий Швец

сменил номер телефона, имя и фамилию

14.06

2 не оплачено

2 пробное занятие

2 не придут

11 из 20

Ученик Дмитрий Швец

сменил номер телефона, имя и фамилию

14.06

2 не оплачено

2 пробное занятие

2 не придут

11 из 20

Ученик Дмитрий Швец

сменил номер телефона, имя и фамилию

14.06

2 не оплачено

2 пробное занятие

2 не придут

11 из 20

17:00

ул. Иванова, д. 7

2 не оплачено

2 пробное занятие

2 не придут

11 из 20

18:00

ул. Иванова, д. 7

2 не оплачено

2 пробное занятие

2 не придут

11 из 20

Среда 15 июня

25 занятий

Следующее занятие

12:00

ул. Иванова, д. 7

2 не оплачено

2 пробное занятие

2 не придут

11 из 20

12:00

ул. Иванова, д. 7

2 не оплачено

2 пробное занятие

2 не придут

11 из 20

					ДП 6226.07.000 ЕК						
					Креслення вигляду екранних форм	Літера		Маса		Масштаб	
						Аркуш 9		Аркушів10			
						КПІ ім. Ігоря Сікорського					
Зм.	Арк.	№ документа	Підпис	Дата	Система інформаційної підтримки навчальних курсів	Каф. АСОІУ					
Розробив	Халін О.І.					Гр. ІС-62					
Перевірив	Мажара О.О.										
Консульт.											
Н. кон.	Новінський В.П.										
Затвердив	Мажара О.О.										

ДП 6226.10.000 ЕК

The image is a composite of two parts. The top part is a screenshot of a web application for course management. It features a multi-view interface with tabs for 'Мое расписание' (My schedule), 'Текущее занятие' (Current lesson), 'Уведомления' (Notifications), and 'Меню' (Menu). The 'Мое расписание' view shows a calendar for June and July with lesson slots. The 'Текущее занятие' view shows details for a lesson on Tuesday, June 14, including the time (12:00), address (ул. Иванова, д. 7), and duration (1 hour 30 minutes). It also lists students and their payment status (e.g., 'Максим Иванов не оплачено 100 ₴'). The 'Уведомления' view shows notifications about student phone number changes. The bottom part of the image is a scan of a physical document titled 'ДП 6226.07.000 ЕК'. It contains a table for student information with columns for 'Зм.' (Content), 'Арк.' (Page), '№ документа' (Document number), 'Підпис' (Signature), and 'Дата' (Date). The table is filled with student names and signatures. Below the table, there is a section for 'Креслення вигляду екранних форм' (Screenshots of screen forms) and 'Система інформаційної підтримки навчальних курсів' (System of informational support of educational courses). The document also includes a header with the course ID 'ДП 6226.07.000 ЕК' and a footer with the institution name 'КПІ ім. Ігоря Сікорського' and the faculty 'Каф. АСОІУ'.

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ
Кафедра автоматизованих систем обробки інформації і управління

До захисту допущено:
В.о. завідувача кафедри
_____ Олександр ПАВЛОВ
(підпис) (вл.ім'я, прізвище)
“ ” _____ 2020 р.

Дипломний проєкт
на здобуття ступеня бакалавра

**за освітньо-професійною програмою «Інформаційні управляючі системи
та технології»
спеціальності 122 «Комп'ютерні науки та інформаційні технології»**

на тему: «Інформаційна система підтримки діяльності балетної
школи» (комплексна тема).

Підсистема графового інтерфейсу серверної частини
застосунку. Індивідуальна частина № 2

Виконав: студент IV курсу, групи ІС-62

Кравчук Олександр Ігорович
(прізвище, ім'я, по батькові)

_____ (підпис)

Керівник

ст. викл. Мажара Ольга Олександрівна
(посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові)

_____ (підпис)

**Консультант з
графічної
документації**

доц., к.т.н., доц. Новінський Валерій Петрович
(посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові)

_____ (підпис)

Рецензент

доц., к.т.н., доц. Михайлова Ірина Юріївна
(посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові)

_____ (підпис)

Засвідчую, що у цьому дипломному проєкті
немає запозичень з праць інших авторів без
відповідних посилань.

Студент _____
(підпис)

Київ – 2020 року

Пояснювальна записка до дипломного проєкту

на тему: Інформаційна система підтримки діяльності балетної школи.
Підсистема графового інтерфейсу серверної частини застосунку.

Київ – 2020 року

АНОТАЦІЯ

Структура та обсяг роботи. Пояснювальна записка дипломного проєкту складається з 3 розділів, 5 рисунків, 1 таблиці, 20 джерел.

У дипломному проєкті реалізована система підтримки діяльності балетної школи у вигляді програмного інтерфейсу. Клієнтські частини системи реалізовані у вигляді застосунку для веб-браузера та нативного мобільного застосунку. Основна мета створення системи - автоматизація та прискорення шаблонних процесів всередині школи, полегшення комунікації з клієнтами.

У першому розділі «Постановка задачі» описано призначення розробки, цілі та задачі, які повинна вирішувати. розроблена частина застосунку.

У розділі «Програмне та технічне забезпечення» обрано та обґрунтовано набір технологій та засобів для створення застосунку, а також описано вимоги, архітектуру та специфікації функцій застосунку.

Технологічний розділ надає інструкцію користувачу по запуску застосунку і роботі з ним. Також в ньому описані використані методи тестування застосунку.

БАЛЕТНА ШКОЛА, GRAPHQL, СЕРВЕРНИЙ ЗАСТОСУНОК,
КОРИСТУВАЦЬКИЙ ІНТЕРФЕЙС

					ДП ІС-6211.00.002 ПЗ					
		Прізвище	Підпис	Дата						
Розроб.		Кравчук О.І.			Інформаційна система підтримки діяльності балетної школи. Підсистема графового інтерфейсу серверної частини застосунку.	Літ.	Арк.	Аркуші		
							2	102		
Керівн.		Мажара О. О.				КПІ ім. Ігоря Сікорського Каф. АСОІУ ар. ІС-62				
Н. кон.		Новінський В.П.								
Затв.		Павлов О.А.								

ABSTRACT

Structure and scope of work. The explanatory note of the diploma project consists of 3 sections, 5 figures, 1 table, 20 sources.

In the diploma project the system of support of the activity of ballet school in the form of the program interface is realized. Client parts of the system are implemented as a web browser application and a native mobile application. The main purpose of creating the system is to automate and accelerate template processes within the school, to facilitate communication with customers.

The first section "Problem statement" describes the purpose of development, goals, and objectives to be solved by the developed part of the application.

The "Software and Hardware" section selects and substantiates a set of technologies and tools for application development, in addition it describes the requirements, architecture, and specifications of application functions.

The technology section provides instructions to the user on how to run the application and work with it. It also describes the application testing methods used.

BALLET SCHOOL, GRAPHQL, SERVER APPLIANCE, USER INTERFACE

ЗМІСТ

ВСТУП.....	5
1 ПОСТАНОВКА ЗАДАЧІ.....	6
1.1 ПРИЗНАЧЕННЯ РОЗРОБКИ	6
1.2 ЦІЛІ ТА ЗАДАЧІ РОЗРОБКИ.....	6
2 ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ	7
2.1 ЗАСОБИ РОЗРОБКИ.....	7
2.2 ВИМОГИ ДО ТЕХНІЧНОГО ЗАБЕЗПЕЧЕННЯ	10
2.2.1 Загальні вимоги	10
2.3 АРХІТЕКТУРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	10
2.3.1 Діаграма класів.....	10
2.3.2 Діаграма компонентів.....	11
2.3.3 Діаграма варіантів використання.....	11
2.3.4 Діаграма послідовності	11
2.3.5 Діаграма розгортання.....	11
2.3.6 Специфікація функцій.....	11
Висновок до розділу	14
3 ТЕХНОЛОГІЧНИЙ РОЗДІЛ.....	16
3.1 КЕРІВНИЦТВО КОРИСТУВАЧА.....	16
3.2 ВИПРОБУВАННЯ ПРОГРАМНОГО ПРОДУКТУ.....	18
3.2.2 Запуск тестів	21
Висновок до розділу	22
ЗАГАЛЬНІ ВИСНОВКИ.....	23
ПЕРЕЛІК ПОСИЛАНЬ.....	24
ДОДАТОК А.....	26

ВСТУП

У наш час все більше і більше підприємств намагається автоматизувати рутинні процеси всередині. Автоматизація процесів дозволяє суттєво скоротити затрати коштів, а також прискорити виконання роботи, адже програмам не потрібні перерви і сон. За допомогою баз даних і інструментів для роботи з ними можна отримувати звітність про роботу системи у автоматичному режимі.

У даній роботі було розроблено серверний застосунок, який призначений автоматизувати рутинні процеси всередині балетної школи. Серверний застосунок надає програмний інтерфейс, який може бути використаний будь якою програмою. Це може бути веб застосунок, мобільний нативний застосунок або навіть десктопне застосування. При такому підході, коли клієнтська та серверна частина є різними застосунками, ці програми можуть розроблятись паралельно різними розробниками і можуть бути написаними на різних мовах і лише використовувати спільну мову запитів до інтерфейсу користувача.

Система зберігає дані про роботу школи у реляційній базі даних, що дозволяє отримувати інформацію за будь який період роботи, агрегувати та фільтрувати дані, отримувати статистику. Всі ці можливості бази даних дозволяють легко проводити складні обрахунки, які вручну можуть проводитись днями.

1 ПОСТАНОВКА ЗАДАЧІ

1.1 Призначення розробки

Призначення підсистеми - забезпечити середовище для зберігання інформації про роботу балетної школи, надати для клієнтської частини застосунку можливість отримувати дані та управляти ними за допомогою програмного інтерфейсу, який доступний на віддаленому сервері.

1.2 Цілі та задачі розробки

Метою є розробка клієнтського застосунку для забезпечення ефективної взаємодії учнів та педагогів балетної школи, ведення журналів відвідування та особистих абонементів, організації занять та налагодження зворотного зв'язку.

Для забезпечення мети система повинна реалізовувати такі цілі:

- Надавати клієнтській частині інтерфейс для отримання та модифікації даних застосунку;
- Зберігати дані, що відносяться до системи в постійному сховищі та мати можливість керувати ними;
- Контролювати доступ до даних через інтерфейс, надаючи клієнтам лише той набір функціональності, які вони повинні мати.

Додатковою вимогою була ефективна організація оновлень застосунку. Доставка оновленої кодової бази повинна проводитись з мінімальним часом простою, щоб клієнтська частина не очікувала відповіді і не отримувала помилок.

					ДП ІС-6211.00.002 ПЗ	Арк.
						6
Змн.	Арк.	№ докум.	Підпис	Дата		

2 ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Засоби розробки

Для розробки бекенд частини застосунку було застосоване інтегроване середовище розробки JetBrains Pycharm[1]. Контроль версій здійснювався за допомогою системи git[2], а репозиторій з програмним кодом розміщений на Gitlab[3].

Основною мовою програмування у бекенд частині було обрано python[4]. Вона надає наступні необхідні можливості для розробки даного проекту:

- Великий вибір внутрішніх і сторонніх бібліотек для вирішення різного роду задач[4];
- Простий і читабельний синтаксис, який дозволяє швидко описувати необхідну логіку фокусуючись на задачі, а не на синтаксисі[5];
- Простота у встановленні та налаштуванні[6];
- Мультипарадигменість;
- Кроссплатформеність;
- Наявність обширної документації для мови програмування і її внутрішніх бібліотек;
- Велика спільнота розробників. Це дозволяє легше і швидше знаходити рішення проблем[7];
- Динамічна типізація. Дозволяє істотно зменшити час написання програми з рахунок зменшення надійності коду, що для малих та середніх проектів не критичним. Важливі частини коду можуть бути опціонально типізованими і мати високу надійність.

Для запуску застосунку використовується система контейнеризації Docker[8], яка забезпечує ізольоване середовище для запуску і збірки

					ДП ІС-6211.00.002 ПЗ	Арк.
						7
Змн.	Арк.	№ докум.	Підпис	Дата		

застосунків. Для локального управління контейнерами застосовується утиліта docker-compose[9], яка за допомогою мінімальної конфігурації дозволяє описати процес розгортання застосунку для локальної розробки.

Для управління docker контейнерами у production середовищі застосовується утиліта docker swarm[10], яка на відміну від docker-compose може забезпечити мінімальний час простою при доставці оновленої кодової бази у production середовище, а також дозволяє легко горизонтально масштабувати застосунок.

Для автоматизації процесу тестування та доставки коду в production середовищі застосовується GitlabCI[11]. За допомогою цієї системи весь код проходить перевірку аналізаторами коду, а також проходить всі тести в автоматичному режимі. При доставці коду у production середовище - GitlabCI окрім тестування та аналізу коду робить збірку всього проекту та будує готові Docker контейнери. Після цього підключається до сервера та ініціалізує впровадження нової версії застосунку.

В якості постійного сховища для даних системи було обрано реляційну базу даних PostgreSQL[12]. Основна перевага цієї бази серед інших - значна кількість добре підтримуваних бібліотек та драйверів для мови Python, зокрема у цій роботі застосовується бібліотека psycopg[13]. Окрім цього база має наступні переваги:

- Велика і активна спільнота розробників як самої бази даних, адже це open source проект, так і розробників-користувачів. Як результат рішення значної кількості проблем і складних ситуацій легко можна знайти в інтернеті;
- Повнотекстовий пошук, а також спеціалізовані для цього індекси, що дозволяє побудувати ефективний пошук на сайті не підключаючи сторонні залежності;
- Велика кількість типів полів у тому числі json та array.
- Різні види реплікації для різних потреб;

					ДП ІС-6211.00.002 ПЗ	Арк.
						8
Змн.	Арк.	№ докум.	Підпис	Дата		

- Легке розгортання в Docker контейнері.

Для зберігання сесій користувача, а також для кешування даних застосовується Redis[14] сховище пар ключ-значення. Значення в сховищі зберігаються в оперативній пам'яті, що надає надзвичайно швидкий доступ до даних. Оскільки сесії користувача і кеш - це ті речі, доступ до яких виконується часто, зберігання цих даних у такому сховищі значно прискорює роботу застосунку ціною оперативної пам'яті. У разі потреби Redis може забезпечити довговічність зберігання даних. Для доступу до Redis із застосунку застосовується бібліотека aioredis[15]

Redis реалізує різні типи даних для значень, а також ефективні методи для роботи з ними, а завдяки механізму pub/sub сховище може бути використане ще й як брокер повідомлень. З великою спільнотою розробників і значною кількістю бібліотек для роботи зі сховищем - рішення до більшості проблем вже готове. Як і PostgreSQL, Redis легко розгортається в Docker.

Для надання інтерфейсу клієнтам було застосовано відносно нову мову запитів GraphQL[16]. На відміну від класичного REST[17] інтерфейсу у графі описуються об'єкти та зв'язки між ними, а не конкретні точки доступу до об'єктів. Також GraphQL має наступні переваги:

- Мова дозволяє чітко визначити набір необхідних клієнту полів, таким чином сервер поверне лише їх і не витратить час на визначення непотрібних полів;
- Одним запитом можна отримати кілька залежних об'єктів;
- Система типів. За допомогою системи типів клієнти можуть бути впевненими, що отримають дані необхідного типу, а також можуть генерувати типи відповідно до графових типів, що може підвищити надійність клієнтського коду;
- Потужний інструмент GraphQL який дозволяє швидко переглянути схему графу і тестувати запити;

- Легке оновлення API без версій. Граф дозволяє додавати нові поля в об'єкти не ламаючи при цьому існуючі клієнтські запити. Старі поля можуть бути поміченими як deprecated, що не зламає існуючі запити, але клієнти бачитимуть, що це поле більше не актуальне.

Для реалізації графового інтерфейсу в застосунку було використано бібліотеку graphene[19], у якій графові об'єкти представляються як класи, а атрибутами графового об'єкта є власне атрибути класу. Значення для атрибутів визначаються за допомогою класових методів. Кожен класовий метод асинхронний, що значно прискорює отримання результату. Для доступу до графу використовується мікрофреймворк aiohttp[20].

2.2 Вимоги до технічного забезпечення

2.2.1 Загальні вимоги

Мінімальні необхідні характеристики комп'ютера для роботи застосунку:

- 1 ГБ або більше оперативної пам'яті;
- 10 ГБ або більше вільного місця на жорсткому диску;
- Встановлена операційна система linux.

2.3 Архітектура програмного забезпечення

2.3.1 Діаграма класів

Діаграма класів описує схему графового інтерфейсу застосунку, кожен клас є графовим об'єктом у якому атрибути є об'єктами представлені атрибутом класу. Також у деяких класах присутні методи, які дозволяють визначити деякі поля за певною логікою, яка описується у методі. Діаграма класів наведена у графічних матеріалах.

2.3.2 Діаграма компонентів

На діаграмі компонентів показано серверну та клієнтські частини, а також їх внутрішні складові. Діаграма представлена у графічних матеріалах.

2.3.3 Діаграма варіантів використання

На діаграмі вказані типи користувачів системи (актори) а також набір дій , які вони можуть виконувати і які серверна частина застосунку повинна реалізувати для клієнтів. Діаграма наведена у графічних матеріалах.

2.3.4 Діаграма послідовності

Діаграми послідовності описують процеси авторизації існуючого користувача та реєстрацію нового. Діаграми наведені у графічних матеріалах.

2.3.5 Діаграма розгортання

На діаграмі показано обчислювальні вузли системи, а також програмні компоненти, що виконуються на цих вузлах. Діаграма наведена у графічних матеріалах.

2.3.6 Специфікація функцій

Специфікація основних логічних функцій, описана в таблиці 2.1:

Таблиця 2.1 – Перелік основних функцій

Назва	Примітка
create_app()	Ініціалізація застосунку.
database()	Ініціалізація підключення до бази даних PostgreSQL
redis()	Ініціалізація підключення до сховища Redis

Продовження таблиці 2.1

get_user()	Отримує інформацію про авторизованого користувача з Redis та БД
authenticate()	Намагається авторизувати користувача за переданими номером телефону та паролем. Генерує помилку у разі невдачі.
social_authenticate()	Намагається авторизувати користувача за токеном з соціальної мережі
create_user()	Створює нового користувача у базі даних
change_password()	Змінює пароль користувача і оновлює дані в БД
create_social_auth()	Зберігає прив'язку користувача до соц. мережі
get_social_profile()	Отримує загальнодоступні дані із соц. мережі про користувача
load_halls	Завантажує інформацію про зали по ідентифікаторам залів.
load_halls_locations	Завантажує локації у яких знаходяться зали по ідентифікаторам залів
load_locations	Завантажує інформацію про локації по ідентифікаторам локацій
load_users	Завантажує профіль користувача з БД по ідентифікатору користувача
load_lesson_templates	Завантажує усі шаблони (розклад) занять з БД
load_lessons	Завантажує інформацію про задану кількість уроків по заданому періоду із БД
load_lessons_period_aggregation	Завантажує та агрегує інформацію про уроки по тижням та місяцям
load_nearest_lesson	Завантажує з БД найближче заняття користувача
load_current_lesson	Завантажує з БД заняття яке відвідує або проводить користувач
load_lesson_visitors	Завантажує з БД відвідувачів уроку по ідентифікатору уроку

Продовження таблиці 2.1

subscribe_user_on_lesson	Записує користувача на урок
create_lessons_from_template	Створює заняття з шаблону (розкладу) уроку.
toggle_payment	Перемикає оплату відвідувача уроку
toggle_visit	Перемикає відвідування відвідувача уроку
move_lesson_visit	Переносить відвідування уроку
freeze_visit_period	Заморожує заданий період відвідування та переносить заняття з цього періоду
logout	Вихід користувача з системи

Висновок до розділу

У даному розділі описано технології, які застосовувались для розробки бекенд частини застосунку. Обґрунтовано доцільність їх застосування та переваги. Обрані ключові технології в застосунку:

- JetBrains Pycharm - інтегроване середовище для розробки застосунку;
- GIT - система контролю версій у проекті;
- Gitlab - віддалений git репозиторій для зберігання кодової бази;
- Python - мова програмування для бекенд частини застосунку;
- PostgreSQL - база даних для постійного зберігання даних застосунку;
- Redis - сховище сесій користувачів та кеш;
- Gitlab CI - інструмент для автоматизації тестування та доставки кодової бази у production середовище;
- Docker - система контейнеризації, середовище у якому запускається застосунок;
- docker-compose - система управління контейнерами для локальної розробки. Забезпечує легкий запуск застосунку з усіма залежностями локально;
- docker-swarm - система управління контейнерами для production середовища. Забезпечує доставку оновленої кодової бази на production середовище з низьким часом простою, а також надає можливості по вертикальному масштабуванню;
- GraphQL - мова запитів до інтерфейсу застосунку.

Було описано мінімальні необхідні технічні вимоги для запуску застосунку.

					ДП ІС-6211.00.002 ПЗ	Арк.
						14
Змн.	Арк.	№ докум.	Підпис	Дата		

Описано схему бази даних, а також наведено діаграми взаємодії компонентів. Надано специфікацію ключових функцій застосунку.

					ДП ІС-6211.00.002 ПЗ	Арк.
						15
Змн.	Арк.	№ докум.	Підпис	Дата		

3 ТЕХНОЛОГІЧНИЙ РОЗДІЛ

3.1 Керівництво користувача

У даному розділі описано як запускати, а також використовувати бекенд частину застосунку.

Початок роботи

Для запуску застосунку потрібно виконати наступну команду: `docker-compose up ballet_school_app`

Після виконання команди можна відкрити вбудований у застосунок інструмент `graphql`, перейшовши за посиланням <http://localhost:8080/api/graph?query={}>.

На рисунку 3.1 показано початкове вікно `graphql`:

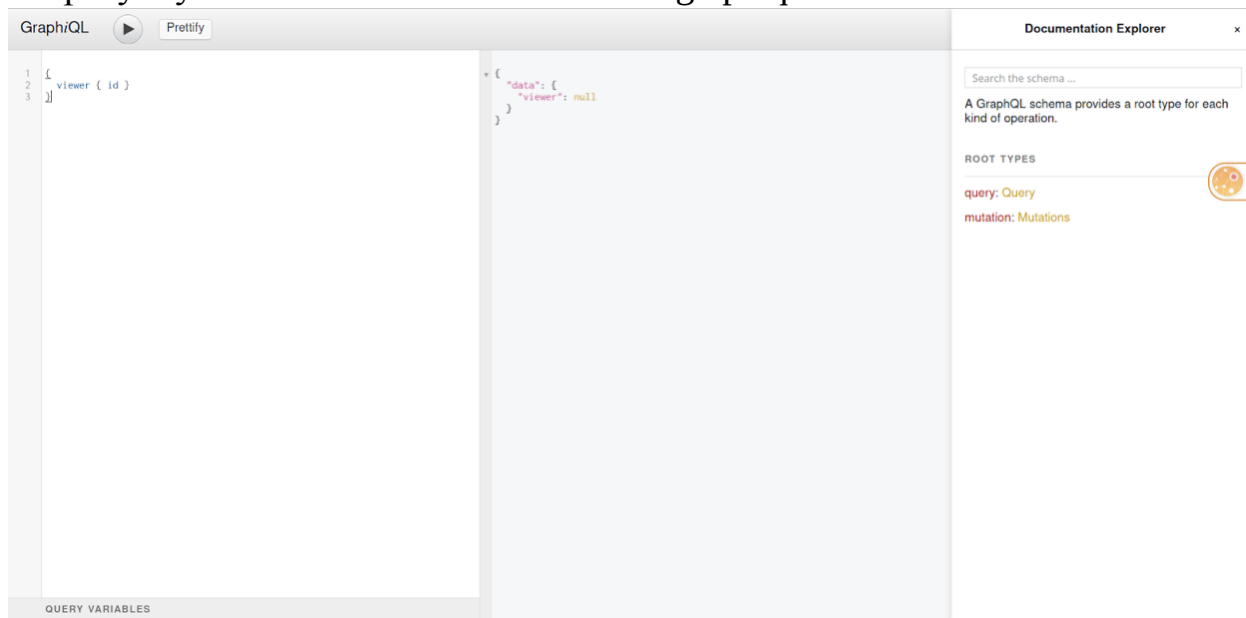


Рисунок 3.1 – Початкове вікно `graphql`

Огляд доступних запитів та мутацій, а також пошук

У блоці `Documentation Explorer` можна переглянути доступні мутації та запити, які надає інтерфейс, а також переглянути детальну інформацію про ці запити та мутації натиснувши на них. На рисунку 3.2 зображено блок `Documentation Explorer`.

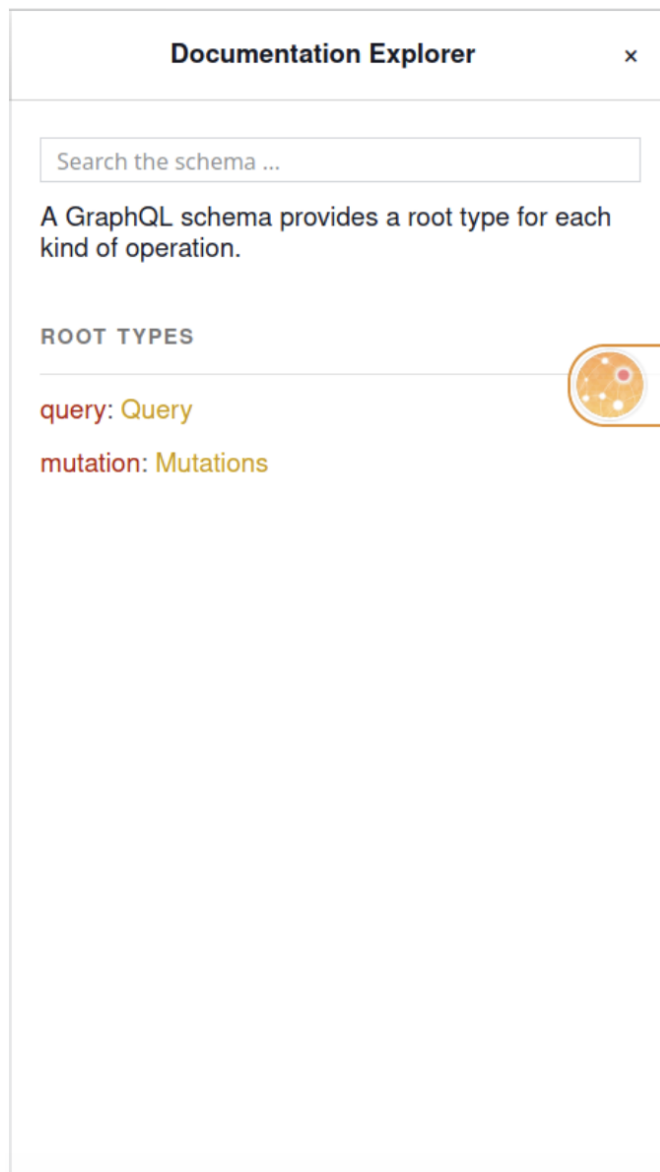


Рисунок 3.2 – блок Documentation Explorer

Тестування запитів

Протестувати як працює graphql запит можна прямо у вікні GraphiQL, для цього можна ввести його у редактор і натиснути на кнопку виконання. На рисунку 3.3 наведено запит і результат його виконання у вікні GraphiQL.

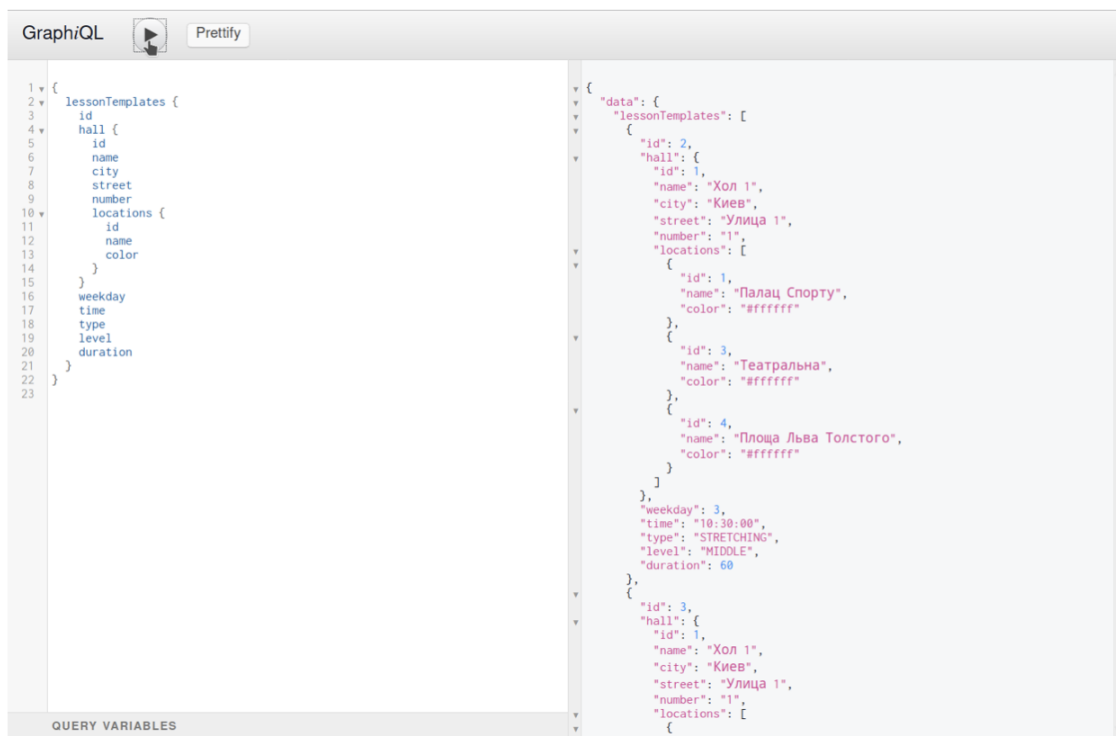


Рисунок 3.3 – GraphQL запит та результат його виконання

3.2 Випробування програмного продукту

Задля забезпечення клієнтів надійним інтерфейсом для взаємодії при розробці застосунку було вирішено покрити юніт-тестами запити та мутації які реалізовані графовим інтерфейсом.

Для тестування запитів створюється фейковий (імітований) запит клієнта і перевіряється співпадіння даних отриманих від сервера і очікуваних даних. Тест вважається пройденим, якщо дані співпадають. На рисунку 3.4 показано оформлення імітованої мутації даних, а на рисунку 3.5 оформлення імітованого запиту даних.

```

250 @pytest.mark.asyncio
251 async def test_mutate_book_unauthenticated(db, fake_request, scheme, loop):
252     PHONE_NUMBER = '+380123456789'
253     VALID_PASSWORD = 'qwerty123'
254     INVALID_PASSWORD = 'zxcvbn567'
255     query = """
256     mutation {
257         authenticate(login: "%s", password: "%s") {
258             isSuccess
259             errors {
260                 field
261                 errors {
262                     code
263                     message
264                 }
265             }
266         }
267     }
268     """
269
270     async with fake_request.app['db'].acquire():
271         await create_user(
272             phone_number=PHONE_NUMBER,
273             password=VALID_PASSWORD
274         )
275
276     # Test authentication with valid credentials
277     executed = await execute_scheme(
278         scheme,
279         query % (PHONE_NUMBER, VALID_PASSWORD),
280         fake_request
281     )
282     assert executed["data"]["authenticate"]["isSuccess"] is True
283
284     # Test authentication with invalid credentials
285     executed = await execute_scheme(
286         scheme,
287         query % (PHONE_NUMBER, INVALID_PASSWORD),
288         fake_request
289     )
290     assert executed["data"]["authenticate"]["isSuccess"] is False
291     assert (
292         executed["data"]["authenticate"]["errors"][0]['errors'][0]['code'] ==
293         'INVALID_CREDENTIALS_ERROR'
294     )

```

Рисунок 3.4 – Тест мутації авторизації

```

297 @pytest.mark.asyncio
298 async def test_lesson_templates_query(
299     db, fake_request, scheme, loop, lesson_templates_fixture,
300 ):
301     query = """
302     {
303         lessonTemplates {
304             id
305             hall {
306                 id
307                 name
308                 city
309                 street
310                 number
311             }
312             weekday
313             time
314             type
315             level
316             duration
317         }
318     }
319     """
320     executed = await execute_scheme(scheme, query, fake_request)
321
322     first_template = executed["data"]["lessonTemplates"][0]
323     assert first_template["id"] == 1
324     assert first_template["weekday"] == 0
325     assert first_template["time"] == "10:25:00"
326     assert first_template["type"] == "PRIMARY"
327     assert first_template["level"] == "MIDDLE"
328     assert first_template["duration"] == 60
329
330     hall = first_template['hall']
331     assert hall["id"] == 1
332     assert hall["name"] == 'Зал 1'
333     assert hall["city"] == 'Kyiv'
334     assert hall["street"] == 'Some str'
335     assert hall["number"] == '25'

```

Рисунок 3.5 – Тест на отримання розкладу занять

Таким чином були протестовані наступні мутації:

- Авторизація користувача;
- Реєстрація користувача через номер телефону;
- Реєстрація користувача через токен соц. мережі;
- Відмічання оплати;
- Відмічання відвідування;

					ДП ІС-6211.00.002 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		20

- Запис на заняття.

Та запити:

- Розклад занять;
- Уроки викладача;
- Інформація про авторизованого користувача;
- Детальна інформація про урок;
- Інформація про нинішнє заняття;
- Інформація про найближче заняття.

3.2.2 Запуск тестів

Всі тести запускаються автоматично системою GitlabCI перед кожною доставкою оновленої кодової бази у production середовище. Якщо в процесі тестування хоч один тест не виконується процес доставки перерветься і потенційно не робоча кодова база не може потрапити у production.

					ДП ІС-6211.00.002 ПЗ	Арк.
						21
Змн.	Арк.	№ докум.	Підпис	Дата		

Висновок до розділу

У даному розділі показано інструкцію для користувача застосунку по його запуску, а також використанню. Описано метод тестування інтерфейсу і показано приклад оформлення тестів. Всі тести оформлені у вигляді імітованого запиту від клієнта на сервер та порівняння отриманого від серверу результату з очікуваним значенням. Також наведено протестований набір запитів та мутацій інтерфейсу і описано процес запуску перевірки тестів.

					ДП ІС-6211.00.002 ПЗ	Арк.
						22
Змн.	Арк.	№ докум.	Підпис	Дата		

ЗАГАЛЬНІ ВИСНОВКИ

У результаті виконання дипломного проекту було розроблено серверний застосунок, який є частиною системи інформаційної підтримки балетної школи. Серверний застосунок надає клієнтам інтерфейс для управління даними.

В процесі виконання роботи відповідно до вимог програмного забезпечення були визначені необхідні програмні інтерфейси, також вхідні та вихідні дані для них.

Згідно поставлених вимог та інтерфейсів було проведено дослідження та визначення набору технологій, які можна застосувати для виконання цього проекту. Додатковою умовою до технологій була швидкість розробки продукту та простота підтримки. В розділі засобів розробки було представлено набір технологій для розробки застосунку, а також обґрунтовано зроблений вибір.

У розділі архітектури застосунку описані класи, компоненти, варіанти використання, які реалізовані у програмному продукті, описано специфікацію функцій, а також наведено діаграму розгортання застосунку в цілому.

У технологічному розділі описано процес запуску серверної частини застосунку, вказано як відкрити інструмент розробника для отримання інформації про інтерфейс графу. Описані ключові елементи інтерфейсу застосунку та показано редактор запитів, за допомогою якого можна швидко протестувати запит до графового інтерфейсу і отримати результат. У частині тестування застосунку показано спосіб тестування інтерфейсу, наведено приклади тестових сценаріїв для запиту та мутації, описано процес автоматичного тестування.

ПЕРЕЛІК ПОСИЛАНЬ

1. Built-in Developer Tools – Features [Електронний ресурс]:[Веб-сайт]. – Електронні дані. – Режим доступу: <https://www.jetbrains.com/pycharm/features/tools.html>;
2. Git Docs [Електронний ресурс]:[Веб-сайт]. – Електронні дані. – Режим доступу: <https://git-scm.com/doc>;
3. GitLab Docs [Електронний ресурс]:[Веб-сайт]. – Електронні дані. – Режим доступу: <https://docs.gitlab.com/>;
4. The Python Standard Library [Електронний ресурс]:[Веб-сайт]. – Електронні дані. – Режим доступу: <https://docs.python.org/3/library/index.html>;
5. The Python Language Reference [Електронний ресурс]:[Веб-сайт]. – Електронні дані. – Режим доступу: <https://docs.python.org/3/reference/index.html>;
6. Python Setup and Usage [Електронний ресурс]:[Веб-сайт]. – Електронні дані. – Режим доступу: <https://docs.python.org/3/using/index.html>;
7. General Python FAQ [Електронний ресурс]:[Веб-сайт]. – Електронні дані. – Режим доступу: <https://docs.python.org/3/faq/general.html>;
8. Docker documentation [Електронний ресурс]:[Веб-сайт]. – Електронні дані. – Режим доступу: <https://docs.docker.com/>;
9. Overview of Docker Compose [Електронний ресурс]:[Веб-сайт]. – Електронні дані. – Режим доступу: <https://docs.docker.com/compose/>;
10. Swarm mode overview [Електронний ресурс]:[Веб-сайт]. – Електронні дані. – Режим доступу: <https://docs.docker.com/engine/swarm/>;
11. GitLab Runner Docs [Електронний ресурс]:[Веб-сайт]. – Електронні дані. – Режим доступу: <https://docs.gitlab.com/runner/>;
12. PostgreSQL 12.3 Documentation [Електронний ресурс]:[Веб-сайт]. – Електронні дані. – Режим доступу: <https://www.postgresql.org/docs/12/index.html>;

13. Psycopg – PostgreSQL database adapter for Python [Електронний ресурс]:[Веб-сайт]. – Електронні дані. – Режим доступу: <https://www.psycopg.org/docs/>;

14. Redis documentation [Електронний ресурс]:[Веб-сайт]. – Електронні дані. – Режим доступу: <https://redis.io/documentation>;

15. aioredis — API Reference [Електронний ресурс]:[Веб-сайт]. – Електронні дані. – Режим доступу: https://aioredis.readthedocs.io/en/v1.3.0/api_reference.html;

16. GraphQL specification [Електронний ресурс]:[Веб-сайт]. – Електронні дані. – Режим доступу: <http://spec.graphql.org/June2018/>;

17. Representational State Transfer (REST) [Електронний ресурс]:[Веб-сайт]. – Електронні дані. – Режим доступу: https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm;

18. GraphiQL & the GraphQL LSP Reference Ecosystem for building browser & IDE tools. [Електронний ресурс]:[Веб-сайт]. – Електронні дані. – Режим доступу: <https://github.com/graphql/graphiql>;

19. Graphene documentation [Електронний ресурс]:[Веб-сайт]. – Електронні дані. – Режим доступу: <https://docs.graphene-python.org/en/latest/>;

20. aiohttp – Server [Електронний ресурс]:[Веб-сайт]. – Електронні дані. – Режим доступу: <https://docs.aiohttp.org/en/stable/web.html>;

Додаток А

Тексти програмного коду

Інформаційна система підтримки діяльності балетної школи.

Підсистема графового інтерфейсу серверної частини застосунку.

(Найменування програми (документа))

DVD-R

(Вид носія даних)

76 арк., 184 800 Кб

(Обсяг програми (документа) , арк.) Кб)

Київ – 2020 року

					ДП ІС-6211.00.002 ПЗ	Арк.
						26
Змн.	Арк.	№ докум.	Підпис	Дата		

```

# app.py
import os
import asyncio
from pathlib import Path
from typing import AsyncGenerator
from functools import partial
from datetime import timedelta

import aiohttp_jinja2
import aiohttp_admin
import aiohttp_security
import aiopg.sa
import aioredis
import aiobotocore
import jinja2
from aiohttp_session import setup
from aiohttp_session.redis_storage import RedisStorage
from aiohttp import (
    web,
    ClientSession
)

from ballet_school.apps.admin.schema import schema
from ballet_school.apps.admin.routes import (
    init_routes as init_admin_routes
)

from ballet_school.apps.admin.security import (
    IdentityPolicy,
    AuthorizationPolicy
)

from ballet_school.graph.dataloaders import UserDataLoader
from ballet_school.middlewares import data_loaders
from ballet_school.routes import init_routes
from ballet_school.utils.common import (
    media_url,
    landing_media_url,
    prettify_phone_number,
    get_secret,
)

from ballet_school.utils.config import app_config
from ballet_school.utils.jinja_globals import (
    js_bundle,
    css_bundle,
    js_asset,
    current_user_processor,
    is_canary_processor,
)

from ballet_school.utils.esputnik import (
    ESputnik,
    DummyESputnik
)

from ballet_school.utils.quota import SmsQuota

```

					ДП ІС-6211.00.002 ПЗ	Арк.
						27
Змн.	Арк.	№ докум.	Підпис	Дата		

```

from ballet_school.utils.telegram import TelegramBot

path = Path(__file__).parent

MEMORY_32_MB = 33554432

app = web.Application(
    debug=os.environ.get('DEBUG') == 'True',
    client_max_size=MEMORY_32_MB
)

def setup_admin() -> None:
    admin = aiohttp_admin._setup(
        app,
        schema=schema,
        db=app['db'],
    )
    init_admin_routes(admin)

ident_policy = IdentityPolicy()
auth_policy = AuthorizationPolicy()
aiohttp_security.setup(
    admin,
    ident_policy,
    auth_policy,
)

app.add_subapp('/admin/api/', admin)

async def init_jinja2(app: web.Application) -> AsyncGenerator[None, None]:
    """
    Initialize jinja2 template for application.
    """
    app['static_root_url'] = app_config['app']['static_url']
    env = aiohttp_jinja2.setup(
        app,
        loader=jinja2.FileSystemLoader(str(path / 'templates')),
        context_processors=(
            current_user_processor,
            is_canary_processor,
            aiohttp_jinja2.request_processor,
        ),
    )

    env.globals['media_url'] = media_url
    env.globals['landing_media_url'] = landing_media_url
    env.globals['prettify_phone_number'] = prettify_phone_number

    if app_config.get('js_bundle') is not None:
        env.globals['js_bundle'] = partial(

```

					ДП ІС-6211.00.002 ПЗ	Арк.
						28
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        js_bundle,
        dev_server_url=app_config['js_bundle']['url']
    )
    env.globals['css_bundle'] = partial(
        css_bundle,
        dev_server_url=app_config['js_bundle']['url']
    )
    env.globals['js_asset'] = partial(
        js_asset,
        dev_server_url=app_config['js_bundle']['url']
    )
else:
    env.globals['js_bundle'] = js_bundle
    env.globals['css_bundle'] = css_bundle
    env.globals['js_asset'] = js_asset

yield

```

```

async def database(app: web.Application) -> AsyncGenerator[None, None]:
    """

```

A function that, when the server is started, connects to postgresql,
and after stopping it breaks the connection (after yield)

```

    config = app_config['postgres']

```

```

    engine = await aiopg.sa.create_engine(**config)
    app['db'] = engine

```

```

    yield

```

```

    app['db'].close()
    await app['db'].wait_closed()

```

```

async def redis(app: web.Application) -> AsyncGenerator[None, None]:
    """

```

A function that, when the server is started, connects to redis,
and after stopping it breaks the connection (after yield)

```

    config = app_config['redis']

```

```

    create_redis = partial(
        aioredis.create_redis,
        fredis://{config["host"]}:{config["port"]}'
    )
    app['create_redis'] = create_redis

```

```

    app['redis'] = await create_redis()

```

```

    yield

```

```
app['redis'].close()
await app['redis'].wait_closed()
```

```
async def session(app: web.Application) -> AsyncGenerator[None, None]:
```

```
    redis = await app['create_redis']()
    storage = RedisStorage(redis)
    setup(app, storage)
```

```
    yield
```

```
    redis.close()
    await redis.wait_closed()
```

```
async def esputnik(app: web.Application) -> AsyncGenerator[None, None]:
```

```
    config = app_config.get('esputnik')
    session = ClientSession()
```

```
    if config:
```

```
        user = config.get('user')
        password = config.get('password')
```

```
    else:
```

```
        user = get_secret('ESPUTNIK_USER')
        password = get_secret('ESPUTNIK_PASSWORD')
```

```
    if app_config['app'].get('dev'):
```

```
        app['esputnik'] = DummyESputnik(
            app=app,
            session=session,
            sms_sender='infomir',
            user=user,
            password=password
        )
```

```
    else:
```

```
        app['esputnik'] = ESputnik(
            app=app,
            session=session,
            sms_sender='infomir',
            user=user,
            password=password
        )
```

```
    yield
```

```
    await session.close()
```

```
async def telegram(app: web.Application) -> AsyncGenerator[None, None]:
```

```
    session = ClientSession()
    bot_token = get_secret('TELEGRAM_NOTIFIER_BOT_TOKEN')
```

					ДП ІС-6211.00.002 ПЗ	Арк.
						30
Змн.	Арк.	№ докум.	Підпис	Дата		

```

app['telegram_bot'] = TelegramBot(
    session=session,
    bot_token=bot_token
)

yield

await session.close()

async def client(app: web.Application) -> AsyncGenerator[None, None]:
    session = ClientSession()
    app['client'] = session

    yield

    await session.close()

async def legacy_data_loaders(
    app: web.Application
) -> AsyncGenerator[None, None]:
    # TODO (Inalkman): remove this legacy loaders
    loaders = {
        'users': UserDataLoader
    }

    for key, loader in loaders.items():
        loaders[key] = loaders[key](app['db'])

    app['data_loaders'] = loaders

    yield

def setup_cors_policy(app: web.Application) -> None:
    import aiohttp_cors

    aiohttp_cors.setup(app, defaults={
        "localhost": aiohttp_cors.ResourceOptions(),
    })

async def storage(app: web.Application) -> AsyncGenerator[None, None]:
    AWS_ACCESS_KEY_ID = get_secret('AWS_ACCESS_KEY_ID')
    AWS_SECRET_ACCESS_KEY = get_secret('AWS_SECRET_ACCESS_KEY')
    AWS_ENDPOINT_URL = get_secret('AWS_ENDPOINT_URL')

    session = aiobotocore.get_session(loop=asyncio.get_event_loop())
    async with session.create_client(
        's3',
        region_name='fra1',

```

					ДП ІС-6211.00.002 ПЗ	Арк.
						31
Змн.	Арк.	№ докум.	Підпис	Дата		

```

aws_access_key_id=AWS_ACCESS_KEY_ID,
aws_secret_access_key=AWS_SECRET_ACCESS_KEY,
endpoint_url=AWS_ENDPOINT_URL
) as client:
    app['storage'] = client

    yield

async def init_app() -> web.Application:
    init_routes(app)
    app.middlewares.extend([
        data_loaders,
    ])

    if app_config['app'].get('dev'):
        setup_cors_policy(app)

    config = app_config['postgres']
    engine = await aiopg.sa.create_engine(**config)
    app['db'] = engine

    app.cleanup_ctx.extend([
        redis,
        session,
        init_jinja2,
        esputnik,
        telegram,
        client,
        legacy_data_loaders,
        storage
    ])

    setup_admin()

    return app

# context.py
from contextvars import ContextVar
from typing import cast

from ballet_school.graph.dataloaders import DataLoaders
from ballet_school.types import ContextVarType

__all__ = ('data_loaders_var',)

data_loaders_var = cast(
    ContextVarType[DataLoaders],
    ContextVar('data_loaders')
```

)

```
# middlewares.py
from typing import (
    Callable,
    Awaitable
)
```

)

```
from aiohttp import web
from aiohttp.web import middleware
```

```
from ballet_school.graph.dataloaders import DataLoaders
from ballet_school import context as ctx
```

```
@middleware
async def data_loaders(
    request: web.Request,
    handler: Callable[[web.Request], Awaitable[web.Response]]
) -> web.Response:
    db_engine = request.config_dict['db']
    ctx.data_loaders_var.set(
        DataLoaders.create_loaders(engine=db_engine)
    )
```

```
response = await handler(request)
return response
```

```
# schema.py
from typing import (
    List,
    Optional
)
import typing as t
```

```
import graphene
from aiopg.sa.result import RowProxy
from graphql import ResolveInfo
```

```
from ballet_school.apps.auth.utils import get_user
from ballet_school.graph.models.user import User
from ballet_school.graph.models.trial_lesson import TrialLesson
from ballet_school.graph.models.lesson_detail import LessonDetail
from ballet_school.graph.objects.lesson_template import LessonTemplate
from ballet_school.graph.objects.mutations import Mutations
from ballet_school.apps.school.dc.lesson_template import (
    LessonTemplate as LessonTemplateDataClass,
)
```

Змн.	Арк.	№ докум.	Підпис	Дата

```

class Query(graphene.ObjectType):
    viewer = graphene.Field(
        User,
        description='Currently authenticated user'
    )

    trial_lessons = graphene.List(
        graphene.NonNull(
            TrialLesson
        ),
        description=(
            'List with available for booking trial lessons, may be null'
            'if authorized user already visited lesson'
        )
    )

    lesson_detail = graphene.Field(
        LessonDetail,
        description=(
            'Detailed information about lesson'
        ),
        lesson_id=graphene.Int(
            required=True
        )
    )

    lesson_templates = graphene.List(
        graphene.NonNull(LessonTemplate),
        required=True,
    )

    async def resolve_viewer(self, info: ResolveInfo) -> RowProxy:
        request = info.context['request']
        return await get_user(request)

    async def resolve_trial_lessons(
        self,
        info: ResolveInfo
    ) -> Optional[List[TrialLesson]]:
        db = info.context['request'].app['db']

        async with db.acquire() as conn:
            return await TrialLesson.resolve_multiple_objects(conn)

    async def resolve_lesson_detail(
        self,
        info: ResolveInfo,
        lesson_id: int
    ) -> Optional[LessonDetail]:
        db = info.context['request'].app['db']

        async with db.acquire() as conn:

```

Змн.	Арк.	№ докум.	Підпис	Дата

```

        return await LessonDetail.resolve_object_by_id(conn, lesson_id)

    async def resolve_lesson_templates(
        self,
        info: ResolveInfo
    ) -> t.Any:
        db = info.context['request'].app['db']

        async with db.acquire() as conn:
            return tuple(
                await LessonTemplateDataClass.get_available_templates(
                    connection=conn
                )
            )

schema = graphene.Schema(query=Query, mutation=Mutations)

# lesson.py
from datetime import date

import graphene
from graphql import ResolveInfo
from sqlalchemy import (
    select,
    func,
    desc,
    false,
    true,
    null
)
from sqlalchemy.engine import RowProxy
from sqlalchemy.sql import Select

from ballet_school.apps.school.tables import (
    Hall,
    LessonVisitor,
    Lesson as LessonTable
)
from ballet_school.graph.objects.visit_analytics import VisitAnalytics
from ballet_school.graph.objects.hall import (
    Hall as GraphHall
)

class Lesson(graphene.ObjectType):

    id = graphene.Int(
        required=True
    )

```

					ДП ІС-6211.00.002 ПЗ	Арк.
						35
Змн.	Арк.	№ докум.	Підпис	Дата		

```

booking_places_count = graphene.Int(
    required=True
)

trial_places_count = graphene.Int(
    required=True
)

hall = graphene.Field(
    'ballet_school.graph.objects.hall.Hall',
    description=(
        'Hall where will be lesson'
    ),
    required=True
)

datetime = graphene.DateTime(
    description=(
        'Date and time when lesson starts'
    ),
    required=True
)

duration = graphene.Int(
    description=(
        'Lesson duration in minutes'
    ),
    required=True
)

visit_analytics = graphene.Field(
    'ballet_school.graph.objects.visit_analytics.VisitAnalytics',
    required=True
)

@staticmethod
def resolve_visit_analytics(
    row: RowProxy,
    _: ResolveInfo
) -> VisitAnalytics:
    return VisitAnalytics(
        in_all=row.visitors_in_all,
        should_present=row.visitors_in_all - row.visitors_will_absent,
        not_paid=row.visitors_not_paid,
        trial_lesson=row.visitors_trial_lesson,
        will_absent=row.visitors_will_absent
    )

@staticmethod
def resolve_hall(
    row: RowProxy,
    _: ResolveInfo

```

					ДП ІС-6211.00.002 ПЗ	Арк.
						36
Змн.	Арк.	№ докум.	Підпис	Дата		

```

) -> GraphHall:
    return GraphHall(
        id=row.hall_id,
        name=row.hall_name,
        city=row.hall_city,
        street=row.hall_street,
        number=row.hall_number
    )

```

```

class LessonConnection(graphene.relay.Connection):

```

```

    class Meta:
        node = Lesson

```

```

# Lesson aggregation
from datetime import date

```

```

import graphene
from aiopg.sa.result import RowProxy
from sqlalchemy import (
    select,
    func,
    desc
)
from sqlalchemy.sql import Select

```

```

from ballet_school.apps.school.tables import Lesson

```

```

class LessonPeriodAggregation(graphene.ObjectType):

```

```

    """

```

```

    Type represent aggregated by some periods (days, weeks ...) lessons data
    """

```

```

    lessons_count = graphene.Int(
        description=(
            'How many lessons in this aggregation period'
        ),
        required=True
    )

```

```

    individual_lessons_count = graphene.Int(
        description=(
            'How many individual lessons in this aggregation period'
        ),
        required=True
    )

```

```

    group_lessons_count = graphene.Int(
        description=(

```

					ДП ІС-6211.00.002 ПЗ	Арк.
						37
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        'How many group lessons in this aggregation period'
    ),
    required=True
)

start_date = graphene.Date(
    description=(
        'Start date of current aggregation period'
    )
)

@staticmethod
def get_coach_lessons_aggregation_query(
    group_by: str,
    coach_id: int,
    first: int,
    after: date = None,
    before: date = None
) -> Select:
    """
    Get first coach lessons aggregation before or after some date.
    When using first with before it means first N aggregations right before
    `before` date (except `before` date)
    """

    truncated_date_expr = func.date_trunc(group_by, Lesson.c.time)
    base_query = (
        select([
            func
            .count(1)
            .label('lessons_count'),

            truncated_date_expr
            .label('start_date')
        ])
        .select_from(Lesson)
        .where(Lesson.c.coach_id == coach_id)
        .group_by(truncated_date_expr)
        .limit(first)
    )

    if before is not None:
        base_query = (
            base_query
            .where(Lesson.c.time < func.date_trunc(group_by, before))
            .order_by(desc(truncated_date_expr))
            .alias('T')
        )

    return (
        select('*')
        .select_from(base_query)
        .order_by(base_query.c.start_date)
    )

```

Змн.	Арк.	№ докум.	Підпис	Дата

if after is None:

after = date.today()

return (

base_query

.where(Lesson.c.time >= func.date_trunc(group_by, after))

.order_by(truncated_date_expr)

)

@staticmethod

def resolve_group_lessons_count(row: RowProxy) -> int:

return row.lessons_count

@staticmethod

def resolve_individual_lessons_count(_: RowProxy) -> int:

return 0

class LessonPeriodAggregationConnection(graphene.relay.Connection):

class Meta:

node = LessonPeriodAggregation

user.py

from datetime import (

datetime,

date

)

from typing import (

Optional,

List

)

import graphene

from aiopg.sa.result import RowProxy

from graphql import ResolveInfo

from sqlalchemy.sql import Select

from sqlalchemy import (

and_,

select

)

from ballet_school.apps.auth.utils import get_user

from ballet_school.apps.school.tables import (

Lesson,

Hall,

LessonVisitor

)

from ballet_school.graph.connection import TwoWayCursorConnection

from ballet_school.graph.enums import (

Змн.	Арк.	№ докум.	Підпис	Дата

```
LessonAggregationPeriodsEnum
)
from ballet_school.graph.models.lesson import (
    LessonConnection,
    Lesson as GraphLesson
)
from ballet_school.graph.models.lesson_aggregation import (
    LessonPeriodAggregationConnection,
    LessonPeriodAggregation
)
from ballet_school.graph.models.lesson_detail import LessonDetail
from ballet_school.graph.utils import login_required
```

```
class User(graphene.ObjectType):
```

```
    id = graphene.Int(
        required=True
    )
```

```
    first_name = graphene.String(
        description='User first name',
        required=True
    )
```

```
    last_name = graphene.String(
        description='User last name',
        nearequired=True
    )
```

```
    phone_number = graphene.String(
        description='User phone number in format: 380123456789',
        required=True
    )
```

```
    avatar_url = graphene.String(
        description='Url on user avatar'
    )
```

```
    is_coach = graphene.Boolean(
        required=True
    )
```

```
    nearest_lesson = graphene.Field(
        LessonDetail
    )
```

```
    lessons = TwoWayCursorConnection(
        LessonConnection,
        cursor_fields=('time',)
    )
```

					ДП ІС-6211.00.002 ПЗ	Арк.
						40
Змн.	Арк.	№ докум.	Підпис	Дата		

```

lessons_period_aggregation = TwoWayCursorConnection(
    LessonPeriodAggregationConnection,
    cursor_fields=('start_date',),
    group_by=graphene.Argument(LessonAggregationPeriodsEnum)
)

```

```

@staticmethod
def get_nearest_lesson_query(
    user_id: int,
    is_coach: bool
) -> Select:
    if is_coach:
        return (
            select(LessonDetail.REQUIRED_DB_FIELDS)
            .select_from(
                Lesson
            ).join(Hall)
        )
    .where(
        and_(
            Lesson.c.coach_id == user_id,
            Lesson.c.time > datetime.now()
        )
    )
    .order_by(Lesson.c.id)
    .limit(1)
)

```

```

return (
    select(LessonDetail.REQUIRED_DB_FIELDS)
    .select_from(
        Lesson
    ).join(Hall)
    .join(
        LessonVisitor,
        LessonVisitor.c.lesson_id == Lesson.c.id
    )
)
.where(
    and_(
        LessonVisitor.c.user_id == user_id,
        Lesson.c.time > datetime.now()
    )
)
.order_by(Lesson.c.id)
.limit(1)
)

```

```

async def resolve_nearest_lesson(
    self,
    info: ResolveInfo
) -> Optional[LessonDetail]:

```

Змн.	Арк.	№ докум.	Підпис	Дата

```

request = info.context['request']
db = request.app['db']
user = await get_user(request)

```

```

if user is None:
    # TODO (Inalkman): add specific error for not authenticated user
    raise Exception('User is not authenticated')

```

```

lesson_detail_query = User.get_nearest_lesson_query(
    user_id=user.id,
    is_coach=user.is_coach
)

```

```

async with db.acquire() as conn:
    lesson_cursor = await conn.execute(lesson_detail_query)
    lesson = await lesson_cursor.first()

```

```

if lesson is None:
    return None

```

```

visit_info_cursor = await conn.execute(
    LessonDetail.get_visitors_info_query(lesson_id=lesson.id)
)
visit_info_rows = await visit_info_cursor.fetchall()

```

```

return LessonDetail.resolve_object_by_rows(
    lesson=lesson,
    visitors=visit_info_rows
)

```

```

@login_required
async def resolve_lessons(
    self,
    info: ResolveInfo,
    first: int,
    before: str = None,
    after: str = None,
) -> List[RowProxy]:
    request = info.context['request']
    app = request.app
    viewer = await get_user(request)

    if viewer is None:
        return []

    before_date = (
        datetime.fromisoformat(before).date()
        if before is not None
        else None
    )
    after_date = (
        datetime.fromisoformat(after).date()

```

					ДП ІС-6211.00.002 ПЗ	Арк.
						42
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        if after is not None
        else None
    )

    query = GraphLesson.get_coach_lessons_query(
        coach_id=viewer.id,
        first=first,
        before=before_date,
        after=after_date
    )

    async with app['db'].acquire() as conn:
        cursor = await conn.execute(query)
        rows = await cursor.fetchall()

    return rows

async def resolve_lessons_period_aggregation(
    self,
    info: ResolveInfo,
    group_by: str,
    first: int,
    before: str = None,
    after: str = None
) -> List[RowProxy]:
    request = info.context['request']
    app = request.app
    viewer = await get_user(request)

    if viewer is None:
        return []

    before_date = (
        date.fromisoformat(before)
        if before is not None
        else None
    )
    after_date = (
        date.fromisoformat(after)
        if after is not None
        else None
    )

    query = LessonPeriodAggregation.get_coach_lessons_aggregation_query(
        group_by=group_by,
        coach_id=viewer.id,
        first=first,
        before=before_date,
        after=after_date
    )

    async with app['db'].acquire() as conn:

```

```
        cursor = await conn.execute(query)
        rows = await cursor.fetchall()

    return rows

# error.py
import graphene

class FieldError(graphene.ObjectType):

    code = graphene.String(
        required=True
    )
    message = graphene.String()

class Error(graphene.ObjectType):

    field = graphene.String()
    errors = graphene.List(
        graphene.NonNull(FieldError)
    )

# hall.py
import graphene
from graphql import ResolveInfo

from ballet_school.apps.school.dc.hall import Hall as HallDataClass
from ballet_school.apps.school.dc.location import Location as LocationDataClass
from ballet_school import context as ctx

class Hall(graphene.ObjectType):

    id = graphene.Int(required=True)

    name = graphene.String(
        description='Verbose name of hall',
        required=True
    )

    city = graphene.String(
        description='City where hall is placed',
        required=True
    )

    street = graphene.String(
        description='Street where hall is placed',
        required=True
```

					ДП ІС-6211.00.002 ПЗ	Арк.
						44
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    )

    number = graphene.String(
        description='Number of building where hall is placed',
        required=True
    )

    locations = graphene.List(
        graphene.NonNull('ballet_school.graph.objects.location.Location'),
        required=True,
    )

    @staticmethod
    async def resolve_locations(
        parent: HallDataClass,
        _: ResolveInfo
    ) -> LocationDataClass:
        data_loaders = ctx.data_loaders_var.get()
        return await data_loaders.hall_locations.load(key=parent.id)

# lesson_template.py
import graphene
from graphql import ResolveInfo

from ballet_school.constants import WEEKDAY_NUMBERS
from ballet_school.graph.enums import (
    LessonTypeEnum,
    SkillLevelEnum
)
from ballet_school.apps.school.dc.hall import Hall as HallDataClass
from ballet_school.apps.school.dc.lesson_template import (
    LessonTemplate as LessonTemplateDataClass
)
from ballet_school import context as ctx

class LessonTemplate(graphene.ObjectType):

    id = graphene.Int(required=True)

    hall = graphene.Field(
        'ballet_school.graph.objects.hall.Hall',
        description='Hall where lessons from this template happens',
        required=True,
    )

    weekday = graphene.Int(
        description='The day when lessons from this template happens',
        required=True
    )

```

					ДП ІС-6211.00.002 ПЗ	Арк.
						45
Змн.	Арк.	№ докум.	Підпис	Дата		

```

time = graphene.String(
    description=(
        'Time when lesson from this template starts.\n'
        '(Format: `09:05` `22:45`)'
    ),
    required=True
)

type = LessonTypeEnum(
    description='Type of lessons created from this template',
    required=True,
)

level = SkillLevelEnum(
    description=(
        'Required skill level to visit lessons created from '
        'this template'
    ),
    required=True,
)

duration = graphene.Int(
    description='Lesson duration in minutes'
)

@staticmethod
def resolve_weekday(
    parent: LessonTemplateDataClass,
    _: ResolveInfo
) -> int:
    return WEEKDAY_NUMBERS[parent.weekday]

@staticmethod
async def resolve_hall(
    parent: LessonTemplateDataClass,
    _: ResolveInfo
) -> HallDataClass:
    data_loaders = ctx.data_loaders_var.get()

    return await data_loaders.halls.load(key=parent.hall_id)

@staticmethod
async def resolve_type(
    parent: LessonTemplateDataClass,
    _: ResolveInfo
) -> LessonTypeEnum:
    return parent.type.value

@staticmethod
async def resolve_level(
    parent: LessonTemplateDataClass,
    _: ResolveInfo

```

) -> LessonTypeEnum:

return parent.skill_level.value

lesson_visitor.py

import graphene

from aiopg.sa.result import RowProxy

from graphql import ResolveInfo

from promise import Promise

class LessonVisitor(graphene.ObjectType):

user = graphene.Field(

'ballet_school.graph.models.user.User',

required=True

)

paid_on_lesson = graphene.Boolean(

description=(

'Is visitor paid for this lesson on lesson'

),

required=True

)

pay_amount = graphene.Int(

description=(

'How much user should pay for this lesson'

),

required=True

)

paid_online = graphene.Boolean(

description=(

'Is visitor paid for this lesson online via website'

),

required=True

)

is_first_visit = graphene.Boolean(

description=(

'Is it trial lesson for this user'

),

required=True

)

moved_to_lesson_id = graphene.Int(

description=(

'Id of lesson on which this user moved his visit'

)

)

					ДП ІС-6211.00.002 ПЗ	Арк.
						47
Змн.	Арк.	№ докум.	Підпис	Дата		

```

visited = graphene.Boolean(
    description=(
        'Is this user was present on lesson'
    )
)

```

```

@staticmethod
async def resolve_user(row: RowProxy, info: ResolveInfo) -> Promise:
    loaders = info.context['request'].app['data_loaders']
    return await loaders['users'].load(row.user_id)

```

```

# location.py
import graphene

```

```

class Location(graphene.ObjectType):

```

```

    id = graphene.Int(required=True)

```

```

    name = graphene.String(
        description='Location name',
        required=True,
    )

```

```

    color = graphene.String(
        description='Color of location',
        required=True,
    )

```

```

# visit_analytics.py
import graphene

```

```

class VisitAnalytics(graphene.ObjectType):

```

```

    """

```

```

    Analytics about lesson visiting

```

```

    """

```

```

    in_all = graphene.Int(
        description=(
            'All people count that subscribed on this lesson'
        )
    )

```

```

    should_present = graphene.Int(
        description=(
            'How many people should be on lesson'
        ),
        required=True
    )

```

					ДП ІС-6211.00.002 ПЗ	Арк.
						48
Змн.	Арк.	№ докум.	Підпис	Дата		

```

not_paid = graphene.Int(
    description=(
        'How many people should pay for this lesson'
    ),
    required=True
)

trial_lesson = graphene.Int(
    description=(
        "How many people first time here and it's their first trial"
        "lesson"
    ),
    required=True
)

will_absent = graphene.Int(
    description=(
        'How many people will be absent because of freeze or movement'
    ),
    required=True
)

```

```

# connection.py
import functools
from typing import (
    List,
    Any,
    Dict,
    Union,
    Sequence,
    Optional
)

import graphene
from graphene.relay.connection import IterableConnectionField
from graphene.utils.thenables import maybe_thenable

```

```

class TwoWayCursorConnection(graphene.ConnectionField):

    def __init__(
        self,
        *args: Any,
        cursor_fields: Optional[Sequence[str]] = None,
        **kwargs: Any
    ):
        if cursor_fields is None:
            cursor_fields = ['id']

        assert len(cursor_fields) == 1, (

```

```

        'Currently only one field based cursor supported'
    )

    self.cursor_fields = cursor_fields
    kwargs.setdefault("before", graphene.String())
    kwargs.setdefault("after", graphene.String())
    kwargs.setdefault("first", graphene.Int())
    super(TwoWayCursorConnection, self).__init__(*args, **kwargs)

def resolve_connection(
    self,
    connection_type: Any,
    args: Dict[str, Any],
    resolved: List[Any]
) -> Union[Dict[str, Any], List[Any]]:

    if isinstance(resolved, connection_type):
        return resolved

    first = args['first'] - 1
    edges = [
        connection_type.Edge(
            node=edge,
            cursor=str(getattr(edge, self.cursor_fields[0]))
        ) for edge in resolved[:first]
    ]

    return {
        "edges": edges,
        "page_info": {
            "start_cursor": edges[0].cursor,
            "end_cursor": edges[-1].cursor,
            "has_next_page": first < len(resolved),
            "has_previous_page": False # TODO: implement feature
        }
    }

def connection_resolver(
    self,
    resolver: Any,
    connection_type: Any,
    root: Any,
    info: Any,
    **args: Any
) -> Any:
    args['first'] = args['first'] + 1
    resolved = resolver(root, info, **args)

    if isinstance(connection_type, graphene.NonNull):
        connection_type = connection_type.of_type

    on_resolve = functools.partial(

```

					ДП ІС-6211.00.002 ПЗ	Арк.
						50
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        self.resolve_connection,
        connection_type,
        args
    )
    return maybe_thenable(resolved, on_resolve)

def get_resolver(self, parent_resolver: Any) -> Any:
    resolver = super(IterableConnectionField, self).get_resolver(
        parent_resolver
    )

    return functools.partial(
        self.connection_resolver,
        resolver,
        self.type
    )

# dataloaders.py
import itertools
import typing as t
from typing import (
    Any,
    List
)
import dataclasses

from aiopg.sa import Engine
from aiodataloader import DataLoader
from sqlalchemy import select

from ballet_school.apps.auth.tables import User
from ballet_school.apps.school.tables import (
    Location as LocationTable,
    Hall,
    HallLocationAssociation
)
from ballet_school.types import RowsProxy
from ballet_school.apps.school.dc.location import Location

__all__ = ('DataLoaders',)

TItem = t.TypeVar('TItem')

class BaseAIODataloader(DataLoader):
    engine: Engine

    def __init__(self, engine: Engine, *args: Any, **kwargs: Any):
        super().__init__(*args, **kwargs)
        self.engine = engine

```

```

def sorted_by_keys(
    self,
    items: t.Iterable[TItem],
    keys: List[int]
) -> t.List[TItem]:
    # TODO (Inalkman): check this method behaviour
    items_dict = {
        key: value for key, value in zip(sorted(set(keys)), items)
    }

    return [items_dict[key] for key in keys]

```

```
class UserDataLoader(BaseAIODataLoader):
```

```

    async def batch_load_fn(self, keys: List[int]) -> RowsProxy:
        async with self.engine.acquire() as conn:
            cursor = await conn.execute(
                select([User])
                .where(User.c.id.in_(keys))
                .order_by(User.c.id)
            )
            rows = await cursor.fetchall()

        return self.sorted_by_keys(items=rows, keys=keys)

```

```
class LocationDataLoader(BaseAIODataLoader):
```

```

    async def batch_load_fn(self, keys: List[int]) -> t.List[Location]:
        """
        :param keys: locations ids
        :return: found locations
        """
        async with self.engine.acquire() as conn:
            cursor = await conn.execute(
                select([LocationTable])
                .where(LocationTable.c.id.in_(keys))
                .order_by(LocationTable.c.id)
            )
            rows = await cursor.fetchall()

        return self.sorted_by_keys(
            items=(Location(**row) for row in rows),
            keys=keys
        )

```

```
class HallDataLoader(BaseAIODataLoader):
```

```

    async def batch_load_fn(self, keys: List[int]) -> RowsProxy:
        async with self.engine.acquire() as conn:

```

					ДП ІС-6211.00.002 ПЗ	Арк.
						52
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        cursor = await conn.execute(
            select([Hall])
            .where(Hall.c.id.in_(keys))
            .order_by(Hall.c.id)
        )
        rows = await cursor.fetchall()

    return self.sorted_by_keys(items=rows, keys=keys)

class HallLocationsDataLoader(BaseAIODataLoader):
    """Loader allows to get `Locations` for `Halls`"""

    async def batch_load_fn(self, keys: List[int]) -> t.List[t.List[Location]]:
        """
        :param keys: halls ids
        :return: found locations for halls
        """
        async with self.engine.acquire() as conn:
            cursor = await conn.execute(
                select([
                    HallLocationAssociation,
                    LocationTable,
                ])
                .select_from(
                    HallLocationAssociation
                    .join(LocationTable)
                )
                .where(HallLocationAssociation.c.hall_id.in_(keys))
                .order_by(HallLocationAssociation.c.hall_id)
            )
            rows = await cursor.fetchall()

            groups = itertools.groupby(
                rows,
                lambda o: getattr(o, 'hall_id')
            )

            groups_dict = {
                hall_id: list(
                    Location(
                        id=location.id,
                        name=location.name,
                        color=location.color,
                    )
                    for location in locations_group
                )
                for hall_id, locations_group in groups
            }

            return [groups_dict.get(int(key), []) for key in keys]

```

```

@dataclasses.dataclass
class DataLoaders:
    halls: HallDataLoader
    hall_locations: HallLocationsDataLoader
    locations: LocationDataLoader
    users: UserDataLoader

    @classmethod
    def create_loaders(cls, engine: Engine) -> 'DataLoaders':
        return cls(
            halls=HallDataLoader(engine),
            hall_locations=HallLocationsDataLoader(engine),
            locations=LocationDataLoader(engine),
            users=UserDataLoader(engine),
        )

# jinja_globals.py
import json
from typing import (
    Dict,
    DefaultDict,
    Any
)
from pathlib import Path
from functools import lru_cache
from collections import defaultdict
from urllib.parse import urljoin

from aiohttp import web

from ballet_school.apps.auth.utils import get_user
from ballet_school.utils.config import app_config
from ballet_school.utils.common import is_canary

JS_BUNDLE_MANIFEST_PATH = (
    Path(__file__).parent.parent /
    'static' /
    'bundle_manifest.json'
)

assert app_config['app']['static_url'][-1] == '/', 'Must end with slash'
JS_BUILD_BASE_URL = app_config['app']['static_url'] + 'jsbuild/'

async def current_user_processor(request: web.Request) -> Dict[str, Any]:
    """Add `current_user` to jinja context"""
    return {'current_user': await get_user(request)}

async def is_canary_processor(request: web.Request) -> Dict[str, Any]:

```

Змн.	Арк.	№ докум.	Підпис	Дата

```

"""Add `is_canary` to jinja context"""
return {'is_canary': await is_canary(request)}

```

```

@lru_cache(maxsize=1)
def get_bundle_manifest() -> Dict[str, Dict[str, str]]:
    with open(JS_BUNDLE_MANIFEST_PATH) as f:
        manifest_data = json.loads(f.read())

    js_data = manifest_data['js']

    manifest: DefaultDict[str, Dict[str, str]] = defaultdict(dict)
    for entry_name, bundles in js_data.items():
        for bundle in bundles:
            if bundle.endswith('.js'):
                manifest[entry_name]['js'] = bundle
            elif bundle.endswith('.css'):
                manifest[entry_name]['css'] = bundle

    return manifest

```

```

@lru_cache(maxsize=128)
def js_bundle(
    entry_name: str,
    dev_server_url: str = None
) -> str:
    """
    :param entry_name: bundle name
    :param dev_server_url: dev server url
    :return: url to bundle
    """

    if dev_server_url is not None:
        return urljoin(
            base=dev_server_url,
            url=JS_BUILD_BASE_URL + f'{entry_name}.bundle.js'
        )

    manifest = get_bundle_manifest()

    return urljoin(
        base=JS_BUILD_BASE_URL,
        url=manifest[entry_name]['js']
    )

```

```

@lru_cache(maxsize=128)
def css_bundle(
    entry_name: str,
    dev_server_url: str = None
) -> str:
    """

```

					ДП ІС-6211.00.002 ПЗ	Арк.
						55
Змн.	Арк.	№ докум.	Підпис	Дата		

```

:param entry_name: bundle name
:param dev_server_url: dev server url
:return: url to bundle
"""

if dev_server_url is not None:
    return urljoin(
        base=dev_server_url,
        url=JS_BUILD_BASE_URL + f'{entry_name}.bundle.css'
    )

manifest = get_bundle_manifest()

return urljoin(
    base=JS_BUILD_BASE_URL,
    url=manifest[entry_name]['css']
)

def js_asset(
    asset_name: str,
    dev_server_url: str = None
) -> str:
    if dev_server_url is not None:
        return urljoin(
            base=dev_server_url,
            url=JS_BUILD_BASE_URL + asset_name
        )
    return urljoin(
        base=JS_BUILD_BASE_URL,
        url=asset_name
    )

# config.py
import argparse
import os
import pathlib
from typing import Any

import trafaret
from trafaret_config import commandline

PATH = pathlib.Path(__file__).parent.parent.parent

CONFIG_FILENAME = os.environ.get('SETTINGS_FILE', 'api.dev.yml')
DEFAULT_CONFIG_PATH = PATH / 'config' / CONFIG_FILENAME

CONFIG_TRAFARET = trafaret.Dict({
    trafaret.Key('app'):

```

```

trafaret.Dict({
    'host': trafaret.String(),
    'port': trafaret.Int(),
    'static_url': trafaret.String(),
    trafaret.Key('dev', optional=True): trafaret.Bool()
}),
trafaret.Key('postgres'):
    trafaret.Dict({
        'host': trafaret.String(),
        'port': trafaret.Int(),
        'user': trafaret.String(),
        'password': trafaret.String(),
        'database': trafaret.String(),
    }),
trafaret.Key('redis'):
    trafaret.Dict({
        'host': trafaret.String(),
        'port': trafaret.Int(),
    }),
trafaret.Key('js_bundle', optional=True):
    trafaret.Dict({
        'url': trafaret.String()
    }),
trafaret.Key('storage', optional=False):
    trafaret.Dict({
        'media_url': trafaret.String(),
    }),
})

def get_config(argv: Any = None) -> Any:
    ap = argparse.ArgumentParser()
    cmdline.standard_argparse_options(
        ap,
        default_config=DEFAULT_CONFIG_PATH,
    )
    options = ap.parse_args(argv)

    return cmdline.config_from_options(options, CONFIG_TRAFARET)

app_config = get_config(['-c', DEFAULT_CONFIG_PATH.as_posix()])

# common.py
import os
import functools
import urllib.parse
from typing import (
    Callable,
    Awaitable,
)

```

Змн.	Арк.	№ докум.	Підпис	Дата

```
from aiohttp import web
from aiohttp_session import get_session

from ballet_school.constants import (
    SessionKeys,
    StorageBuckets,
)
from ballet_school.utils.config import app_config

TFun = Callable[..., Awaitable[web.Response]]

def get_secret(secret_name: str) -> str:
    secret = os.environ.get(secret_name)

    if secret is None:
        with open(f'/run/secrets/{secret_name}') as f:
            secret = f.read().strip()

    return secret

async def is_canary(request: web.Request) -> bool:
    session = await get_session(request)

    return SessionKeys.CANARY_FLAG in session

def media_url(path: str) -> str:
    return urllib.parse.urljoin(
        app_config['storage']['media_url'],
        path
    )

def landing_media_url(path: str) -> str:
    return urllib.parse.urljoin(
        app_config['storage']['media_url'],
        f'{StorageBuckets.LANDING}/{path}'
    )

def prettyfy_phone_number(normalized_phone_number: str) -> str:
    """
    :param normalized_phone_number: in format 380123456789
    :return: pretty phone number in format +380 (12) 345 67 89
    """
    s = normalized_phone_number
    return (
        f'+{s[0:3]} '
    )
```

					ДП ІС-6211.00.002 ПЗ	Арк.
						58
Змн.	Арк.	№ докум.	Підпис	Дата		

```
f({s[3:5]}) '  
f({s[5:7]}) '  
f({s[7:9]}) '  
f({s[9:12]}) '  
)
```

```
def canary_view(resolve: TFun) -> TFun:
```

```
@functools.wraps(resolve)  
async def inner(  
    request: web.Request  
) -> web.Response:  
    if not await is_canary(request):  
        raise web.HTTPFound(location='/')  
  
    return await resolve(request)  
  
return inner
```

```
# auth/tables.py  
import sqlalchemy as sa  
from sqlalchemy import text
```

```
from ballet_school.migrations import metadata  
from ballet_school.apps.auth.enums import SocialAuthProviderEnum
```

```
User = sa.Table(  
    'auth_user',  
    metadata,  
    sa.Column(  
        'id',  
        sa.Integer,  
        primary_key=True,  
        index=True  
    ),  
    sa.Column(  
        'email',  
        sa.String(length=200),  
        nullable=True,  
        unique=True  
    ),  
    sa.Column(  
        'first_name',  
        sa.String(length=24),  
        nullable=False  
    ),  
    sa.Column(  
        'last_name',  
        sa.String(length=24),
```

```
        nullable=False
    ),
    sa.Column(
        'phone_number',
        sa.String(length=12),
        nullable=False,
        unique=True
    ),
    sa.Column(
        'password',
        sa.LargeBinary(length=128),
        nullable=False
    ),
    sa.Column(
        'avatar_url',
        sa.Text,
        nullable=True
    ),
    sa.Column(
        'is_coach',
        sa.Boolean,
        server_default=text('false'),
        nullable=False
    ),
    sa.Column(
        'notes',
        sa.Text,
        server_default=text(''),
        nullable=False
    ),
    sa.Column(
        'is_staff',
        sa.Boolean,
        server_default=text('false'),
        nullable=False
    ),
    sa.Column(
        'is_superuser',
        sa.Boolean,
        server_default=text('false'),
        nullable=False
    ),
)
```

```
UserSocialAuth = sa.Table(
    'user_social_auth',
    metadata,
    sa.Column(
        'id',
        sa.Integer,
        primary_key=True,
```

```
        index=True
    ),
    sa.Column(
        'user_id',
        None,
        sa.ForeignKey('auth_user.id'),
        nullable=False
    ),
    sa.Column(
        'uid',
        sa.String(length=255),
        nullable=False
    ),
    sa.Column(
        'provider',
        sa.Enum(SocialAuthProviderEnum),
        nullable=False
    )
)

# auth/db_utils.py
from typing import (
    Collection,
    Optional
)

import crypto
from aiopg.sa import SAConnection
from aiopg.sa.result import RowProxy
from sqlalchemy import (
    Column,
    select,
    exists
)

from ballet_school.apps.auth.enums import SocialAuthProviderEnum
from ballet_school.apps.auth.tables import (
    User,
    UserSocialAuth
)

async def create_user(
    conn: SAConnection,
    phone_number: str,
    encoded_password: bytes,
    first_name: str,
    last_name: str,
    avatar_url: str = None,
    email: str = None,
    returning: Collection[Column] = None
```

					ДП ІС-6211.00.002 ПЗ	Арк.
						61
Змн.	Арк.	№ докум.	Підпис	Дата		

```

) -> Optional[RowProxy]:
    stmt = User.insert().values(
        phone_number=phone_number,
        password=crypto.hash(encoded_password, crypto.salt()),
        first_name=first_name,
        last_name=last_name,
        avatar_url=avatar_url,
        email=email
    )

    if returning is not None:
        return await conn.execute(
            stmt.returning(*returning)
        )

    return await conn.execute(stmt)

async def create_social_auth(
    conn: SAConnection,
    user_id: int,
    uid: str,
    provider: SocialAuthProviderEnum
) -> None:
    await conn.execute(
        UserSocialAuth.insert().values(
            user_id=user_id,
            uid=uid,
            provider=provider
        )
    )

async def change_user_password(
    conn: SAConnection,
    user_id: int,
    new_password: bytes
) -> None:
    await conn.execute(
        User
        .update()
        .where(User.c.id == user_id)
        .values(password=crypto.hashpw(new_password, crypto.salt()))
    )

async def is_phone_number_in_use(
    conn: SAConnection,
    phone_number: str
) -> bool:
    """
    Check whether phone number used by any user.

```

					ДП ІС-6211.00.002 ПЗ	Арк.
						62
Змн.	Арк.	№ докум.	Підпис	Дата		

`phone\_number` should be normalized to proper check.

"""

```
rows = await conn.execute(
    select(
        exists()
        .where(User.c.phone_number == phone_number)
    )
)
row = await rows.fetchone()

return row[0]
```

auth/utils.py

import random

import string

from typing import (

Optional,

NamedTuple

)

import crypto

from aiohttp import (

web,

ClientSession

)

from aiopg.sa.result import RowProxy

from aiohttp_session import get_session

from ballet_school.apps.auth.exceptions import InvalidCredentialsError

from ballet_school.apps.auth.tables import User

from ballet_school.constants import SessionKeys

MAX_PHONE_NUMBER_LENGTH = 13

MIN_PHONE_NUMBER_LENGTH = 9

USER_KEY = 'user'

class SocialUserProfile(NamedTuple):

id: str

first_name: str

last_name: Optional[str] = None

email: Optional[str] = None

picture_url: Optional[str] = None

def normalize_phone_number(phone_number: str) -> Optional[str]:

"""

Normalize input `phone\_number` to format:

3801234567890

Returns `None` if `phone\_number` invalid

					ДП ІС-6211.00.002 ПЗ	Арк.
						63
Змн.	Арк.	№ докум.	Підпис	Дата		

```

"""

```

```

phone_number_len = len(phone_number)

```

```

if phone_number.startswith('+'):

```

```

    valid_phone_number = (

```

```

        phone_number_len == 13 and

```

```

        phone_number.startswith('+380') and

```

```

        phone_number[1:].isdigit()

```

```

    )

```

```

    if valid_phone_number:

```

```

        return phone_number[1:]

```

```

elif phone_number.isdigit():

```

```

    if phone_number.startswith('380') and phone_number_len == 12:

```

```

        return phone_number

```

```

    elif phone_number.startswith('80') and phone_number_len == 11:

```

```

        return '3' + phone_number

```

```

    elif phone_number.startswith('0') and phone_number_len == 10:

```

```

        return '38' + phone_number

```

```

    return None

```

```

async def get_user(request: web.Request) -> Optional[RowProxy]:

```

```

    """

```

```

    Get currently authenticated user, returns `None` if there is no user
    connected to current session.

```

```

    """

```

```

    user = request.get(USER_KEY)

```

```

    if user is None:

```

```

        session = await get_session(request)

```

```

        user_id = session.get(SessionKeys.USER)

```

```

        if user_id is not None:

```

```

            db = request.config_dict['db']

```

```

            async with db.acquire() as conn:

```

```

                cursor = await conn.execute(

```

```

                    User

```

```

                    .select()

```

```

                    .where(User.c.id == user_id)

```

```

                )

```

```

                user = await cursor.first()

```

```

                request[USER_KEY] = user

```

```

    return user

```

```

async def _force_authenticate(request: web.Request, user_id: int) -> None:

```

```

    session = await get_session(request)

```

```

    session[SessionKeys.USER] = user_id

```

```

async def authenticate(

```

Змн.	Арк.	№ докум.	Підпис	Дата

```

request: web.Request,
phone_number: str,
password: str
) -> None:
    """
    Authenticates user with `phone_number` and `password`,
    raise `InvalidCredentialsError` if credentials invalid.
    """

    normalized_phone_number = normalize_phone_number(phone_number)
    if normalized_phone_number is None:
        raise InvalidCredentialsError

    db = request.config_dict['db']
    async with db.acquire() as conn:
        cursor = await conn.execute(
            User
            .select()
            .with_only_columns([
                User.c.id,
                User.c.password
            ])
            .where(User.c.phone_number == normalized_phone_number)
        )

        user = await cursor.first()

    try:
        credentials_valid = (
            user is not None and
            crypto.check(
                password=password.encode('utf-8'),
                hashed_password=user.password
            )
        )
    except UnicodeEncodeError:
        credentials_valid = False

    if credentials_valid:
        await _force_authenticate(request, user.id)
    else:
        raise InvalidCredentialsError

async def logout(request: web.Request) -> None:
    session = await get_session(request)
    del session[SessionKeys.USER]

async def get_google_user_profile(
    client: ClientSession,
    access_token: str
) -> Optional[SocialUserProfile]:

```

```

response = await client.get(
    url='https://www.googleapis.com/oauth2/v1/userinfo?alt=json',
    headers={
        'Authorization': f'Bearer {access_token}'
    }
)
data = await response.json()

if data.get('id') is None:
    return None

return SocialUserProfile(
    id=data['id'],
    first_name=data['given_name'],
    last_name=data.get('family_name', ''),
    email=data.get('email') if data.get('verified_email') else None,
    picture_url=data.get('picture')
)

async def get_facebook_user_profile(
    client: ClientSession,
    access_token: str
) -> Optional[SocialUserProfile]:
    response = await client.get(
        url='https://graph.facebook.com/me',
        params={
            'fields': 'id,first_name,last_name,email,picture',
            'access_token': access_token
        }
    )
    data = await response.json()

    if data.get('id') is None:
        return None

    return SocialUserProfile(
        id=data['id'],
        first_name=data['first_name'],
        last_name=data.get('last_name', ''),
        email=data.get('email'),
        picture_url=data.get('picture', {}).get('data', {}).get('url')
    )

# school/admin.py
import json
from typing import Dict

from aiohttp import web
from aiohttp_admin import (
    require,

```

Змн.	Арк.	№ докум.	Підпис	Дата

Permissions

```
)
from aiohttp_admin.backends.sa import PGResource
from aiohttp_admin.backends.sa_utils import create_filter
from aiohttp_admin.contrib import models
import trafaret as t
from aiohttp_admin.exceptions import (
    ObjectNotFound,
    JsonValidaitonError
)
from aiohttp_admin.utils import (
    validate_query,
    calc_pagination,
    ASC,
    validate_payload,
    json_response
)
import sqlalchemy as sa
from aiopg.sa.result import RowProxy
from trafaret import Any

from ballet_school.apps.school.tables import (
    Hall as HallTable,
    LessonTemplate as LessonTemplateTable,
    Location as LocationTable,
    HallLocationAssociation as HallLocationAssociationTable
)

from ballet_school import context as ctx

class LocationResource(PGResource):

    def __init__(self, *args: Any, **kwargs: Any):
        super().__init__(*args, **kwargs)

        self._create_validator = t.Any()
        self._update_validator = t.Any()

class Location(models.ModelAdmin):

    class Meta:
        resource_type = LocationResource
        table = LocationTable

class HallResource(PGResource):

    def __init__(self, *args: Any, **kwargs: Any):
        super().__init__(*args, **kwargs)

        self._create_validator = t.Dict({
```

					ДП ІС-6211.00.002 ПЗ	Арк.
						67
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        t.Key('name', optional=True): t.String(),
        t.Key('city', optional=True): t.String(),
        t.Key('street', optional=True): t.String(),
        t.Key('number', optional=True): t.String(),
    }).ignore_extra('*')
self._update_validator = t.Dict({
    t.Key('name', optional=True): t.String(),
    t.Key('city', optional=True): t.String(),
    t.Key('street', optional=True): t.String(),
    t.Key('number', optional=True): t.String(),
}).ignore_extra('*')

async def detail(self, request: web.Request) -> web.Response:
    await require(request, Permissions.view)
    entity_id = request.match_info['entity_id']
    data_loaders = ctx.data_loaders_var.get()

    rec = await data_loaders.halls.load(entity_id)

    if not rec:
        msg = 'Entity with id: {} not found'.format(entity_id)
        raise ObjectNotFound(msg)

    entity = dict(rec)
    # TODO (Inalkman): add more performant data loader
    locations = await data_loaders.hall_locations.load(entity_id)
    locations_ids = [location.id for location in locations]
    entity['locations_ids'] = locations_ids
    return web.json_response(entity)

async def create(self, request: web.Request) -> web.Response:
    await require(request, Permissions.add)
    raw_payload = await request.read()
    data = validate_payload(raw_payload, self._create_validator)
    try:
        parsed = json.loads(raw_payload)
    except ValueError:
        raise JsonValidaitonError('Payload is not json serialisable')

    async with self.pool.acquire() as conn:
        try:
            await conn.begin()
            query = self.table.insert().values(data).returning(
                *self.table.c
            )
            rec = await conn.execute(query)
            row = await rec.first()

            locations_ids = parsed.get('locations_ids')
            if locations_ids is not None:
                query = HallLocationAssociationTable.insert().values([
                    {'hall_id': row.id, 'location_id': location_id}

```

Змн.	Арк.	№ докум.	Підпис	Дата

```

        for location_id in parsed['locations_ids']
    )
    await conn.execute(query)
except Exception as e:
    await conn.rollback()
    raise e
else:
    await conn.commit()

entity = dict(row)
return json_response(entity)

async def update(self, request: web.Request) -> web.Response:
    await require(request, Permissions.edit)
    data_loaders = ctx.data_loaders_var.get()
    entity_id = request.match_info['entity_id']
    raw_payload = await request.read()
    data = validate_payload(raw_payload, self.update_validator)
    try:
        parsed = json.loads(raw_payload)
    except ValueError:
        raise JsonValidaitonError('Payload is not json serialisable')

    locations_ids = parsed.get('locations_ids', [])
    locations_ids_set = set(locations_ids)
    async with self.pool.acquire() as conn:
        async with conn.begin():
            query = self.table.select().where(self.f_pk == entity_id)
            row = await conn.execute(query)
            rec = await row.first()
            if not rec:
                msg = 'Entity with id: { } not found'.format(entity_id)
                raise ObjectNotFound(msg)

            db_hall_locations = await data_loaders.hall_locations.load(
                entity_id
            )
            db_locations_ids = set(
                hall_location.location_id
                for hall_location in db_hall_locations
            )

            connections_to_add = locations_ids_set - db_locations_ids
            connections_to_remove = db_locations_ids - locations_ids_set

            row = await conn.execute(
                self.table.update()
                .values(data)
                .returning(*self.table.c)
                .where(self.f_pk == entity_id)
            )
            rec = await row.first()

```

Змн.	Арк.	№ докум.	Підпис	Дата

```

if connections_to_add:
    query = HallLocationAssociationTable.insert().values([
        {'hall_id': rec.id, 'location_id': location_id}
        for location_id in connections_to_add
    ])
    await conn.execute(query)
if connections_to_remove:
    query = HallLocationAssociationTable.delete().where(
        HallLocationAssociationTable.c.location_id.in_(
            connections_to_remove
        )
    )
    await conn.execute(query)

entity = dict(rec)
entity['locations_ids'] = locations_ids
return web.json_response(entity)

```

```
class Hall(models.ModelAdmin):
```

```
class Meta:
```

```

    resource_type = HallResource
    table = HallTable

```

```
class LessonTemplateResource(PGResource):
```

```

def __init__(
    self,
    db: Any,
    table: Any,
    primary_key: Any = 'id',
    url: Any = None
):
    super(PGResource, self).__init__(
        primary_key=primary_key,
        resource_name=url
    )
    self._db = db
    self._table = table
    self._primary_key = primary_key
    self._pk = table.c[primary_key]

    self._create_validator = t.Any()
    self._update_validator = t.Any()

```

```

def prepare_object(self, obj: RowProxy) -> Dict[str, Any]:
    prepared_obj = dict(obj)

```

```
if obj.skill_level:
```

					ДП ІС-6211.00.002 ПЗ	Арк.
						70
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        prepared_obj['skill_level'] = obj.skill_level.name

    if obj.type:
        prepared_obj['type'] = obj.type.name

    if obj.weekday:
        prepared_obj['weekday'] = obj.weekday.name

    if obj.time:
        prepared_obj['time'] = str(obj.time)

    return prepared_obj

async def list(self, request: web.Request) -> web.Response:
    await require(request, Permissions.view)
    columns_names = list(self.table.c.keys())
    q = validate_query(request.query, columns_names)
    paging = calc_pagination(q, self.primary_key)

    filters = q.get('_filters')
    async with self.pool.acquire() as conn:
        if filters:
            query = create_filter(self.table, filters)
        else:
            query = self.table.select()
        count = await conn.scalar(
            sa.select([sa.func.count()])
            .select_from(query.alias('foo')))

        sort_dir = sa.asc if paging.sort_dir == ASC else sa.desc
        cursor = await conn.execute(
            query
            .offset(paging.offset)
            .limit(paging.limit)
            .order_by(sort_dir(paging.sort_field)))

        recs = await cursor.fetchall()

        entities = list(map(self.prepare_object, recs))

    headers = {'X-Total-Count': str(count)}
    return web.json_response(entities, headers=headers)

async def create(self, request: web.Request) -> web.Response:
    await require(request, Permissions.add)
    raw_payload = await request.read()
    data = validate_payload(raw_payload, self.create_validator)

    async with self.pool.acquire() as conn:
        query = self.table.insert().values(data).returning(*self.table.c)
        rec = await conn.execute(query)
        row = await rec.first()

```

					ДП ІС-6211.00.002 ПЗ	Арк.
						71
Змн.	Арк.	№ докум.	Підпис	Дата		

```
await conn.execute('commit;')
```

```
entity = self.prepare_object(row)
return json_response(entity)
```

```
async def detail(self, request: web.Request) -> web.Response:
```

```
await require(request, Permissions.view)
entity_id = request.match_info['entity_id']
async with self.pool.acquire() as conn:
    query = self.table.select().where(self._pk == entity_id)
    resp = await conn.execute(query)
    rec = await resp.first()
```

```
if not rec:
    msg = 'Entity with id: {} not found'.format(entity_id)
    raise ObjectNotFound(msg)
```

```
entity = self.prepare_object(rec)
return json_response(entity)
```

```
async def update(self, request: web.Request) -> web.Response:
```

```
await require(request, Permissions.edit)
entity_id = request.match_info['entity_id']
raw_payload = await request.read()
data = validate_payload(raw_payload, self._update_validator)
```

```
# TODO: execute in transaction?
```

```
async with self.pool.acquire() as conn:
    query = self.table.select().where(self._pk == entity_id)
    row = await conn.execute(query)
    rec = await row.first()
    if not rec:
        msg = 'Entity with id: {} not found'.format(entity_id)
        raise ObjectNotFound(msg)
```

```
row = await conn.execute(
    self.table.update()
    .values(data)
    .returning(*self.table.c)
    .where(self._pk == entity_id))
rec = await row.first()
await conn.execute('commit;')
```

```
entity = self.prepare_object(rec)
return json_response(entity)
```

```
class LessonTemplate(models.ModelAdmin):
```

```
class Meta:
    resource_type = LessonTemplateResource
    table = LessonTemplateTable
```

					ДП ІС-6211.00.002 ПЗ	Арк.
						72
Змн.	Арк.	№ докум.	Підпис	Дата		

```
# school/db_utils.py
from aiopg.sa import SAGConnection
from sqlalchemy import (
    exists,
    select,
    and_
)

from ballet_school.apps.school.tables import (
    LessonVisitor,
    Lesson
)

async def is_user_coach_for_lesson_visitor(
    conn: SAGConnection,
    coach_id: int,
    lesson_visitor_id: int
) -> bool:
    query = (
        select([
            exists()
            .select_from(
                LessonVisitor
                .join(Lesson, LessonVisitor.c.lesson_id == Lesson.c.id)
            )
            .where(
                and_(
                    Lesson.c.coach_id == coach_id,
                    LessonVisitor.c.cid == lesson_visitor_id
                )
            )
        ])
    )

    rows = await conn.execute(query)

    return bool(await rows.fetchone())

# school/tables.py
import sqlalchemy as sa
from sqlalchemy.engine.default import DefaultExecutionContext

from ballet_school.apps.school.enums import (
    Weekdays,
    SkillLevel,
    LessonType
)

from ballet_school.migrations import metadata
```

					ДП ІС-6211.00.002 ПЗ	Арк.
						73
Змн.	Арк.	№ докум.	Підпис	Дата		

```
def hall_name_default(context: DefaultExecutionContext) -> str:
    params = context.get_current_parameters()
    return f'{{params['city']}}, {{params['street'], params['number']}}''
```

```
Location = sa.Table(
    'location',
    metadata,
    sa.Column(
        'id',
        sa.Integer,
        primary_key=True
    ),
    sa.Column(
        'name',
        sa.String(length=64),
    ),
    sa.Column(
        'color',
        sa.String(length=7),
        nullable=False,
        default='#ffbc11',
    )
)
```

```
HallLocationAssociation = sa.Table(
    'halls_locations',
    metadata,
    sa.Column(
        'hall_id',
        None,
        sa.ForeignKey('hall.id', ondelete='CASCADE'),
        nullable=False
    ),
    sa.Column(
        'location_id',
        None,
        sa.ForeignKey('location.id', ondelete='CASCADE'),
        nullable=False
    ),
    sa.UniqueConstraint('hall_id', 'location_id')
)
```

```
Hall = sa.Table(
    'hall',
    metadata,
    sa.Column(
        'id',
        sa.Integer,
        primary_key=True,
```

```
        index=True
    ),
    sa.Column(
        'name',
        sa.String(length=64),
        nullable=False,
        server_default='Зан'
    ),
    sa.Column(
        'city',
        sa.String(length=32),
        nullable=False,
        default='Київ',
        server_default='Київ'
    ),
    sa.Column(
        'street',
        sa.String(length=48),
        nullable=False
    ),
    sa.Column(
        'number',
        sa.String(16)
    )
)

LessonSubscription = sa.Table(
    'lesson_subscription',
    metadata,
    sa.Column(
        'id',
        sa.Integer,
        primary_key=True,
        index=True
    ),
    sa.Column(
        'lessons_count',
        sa.Integer,
        sa.CheckConstraint('lessons_count > 0'),
        nullable=False
    ),
    sa.Column(
        'moves_count',
        sa.Integer,
        sa.CheckConstraint('moves_count >= 0'),
        nullable=False,
        server_default=sa.text('0')
    ),
    sa.Column(
        'freeze_days_count',
        sa.Integer,
```

Змн.	Арк.	№ докум.	Підпис	Дата

```

        sa.CheckConstraint('freeze_days_count >= 0'),
        nullable=False,
        server_default=sa.text('0')
    ),
    sa.Column(
        'price',
        sa.Integer,
        sa.CheckConstraint('price >= 0'),
        nullable=False
    ),
    sa.Column(
        'available',
        sa.Boolean,
        nullable=False,
        server_default=sa.text('false')
    ),
    sa.Column(
        'created_at',
        sa.DateTime,
        nullable=False,
        server_default=sa.func.now()
    ),
    sa.Column(
        'updated_at',
        sa.DateTime,
        nullable=False,
        server_default=sa.func.now(),
        server_onupdate=sa.func.now()
    )
)

```

```

LessonTemplate = sa.Table(
    'lesson_template',
    metadata,
    sa.Column(
        'id',
        sa.Integer,
        primary_key=True,
        index=True
    ),
    sa.Column(
        'hall_id',
        None,
        sa.ForeignKey('hall.id'),
        nullable=False
    ),
    sa.Column(
        'weekday',
        sa.Enum(Weekdays),
        nullable=False
    ),
)

```

```
sa.Column(  
    'skill_level',  
    sa.Enum(SkillLevel),  
    nullable=False  
)  
sa.Column(  
    'type',  
    sa.Enum(LessonType),  
    nullable=False  
)  
  
# Time when lesson starts  
sa.Column(  
    'time',  
    sa.Time,  
    nullable=False  
)  
  
# Lesson duration in minutes  
sa.Column(  
    'duration',  
    sa.Integer,  
    nullable=False,  
    server_default=sa.text('60')  
)  
sa.Column(  
    'coach_id',  
    None,  
    sa.ForeignKey('auth_user.id'),  
    nullable=True  
)  
sa.Column(  
    'booking_places_count',  
    sa.Integer,  
    sa.CheckConstraint('booking_places_count >= 0'),  
    nullable=False  
)  
sa.Column(  
    'trial_places_count',  
    sa.Integer,  
    sa.CheckConstraint('trial_places_count >= 0'),  
    server_default=sa.text('0'),  
    nullable=False  
)  
sa.Column(  
    'available',  
    sa.Boolean,  
    server_default=sa.text('false'),  
    nullable=False,  
    index=True  
)  
)
```

Змн.	Арк.	№ докум.	Підпис	Дата

```

Lesson = sa.Table(
    'lesson',
    metadata,
    sa.Column(
        'id',
        sa.Integer,
        primary_key=True,
        index=True
    ),
    sa.Column(
        'lesson_template_id',
        None,
        sa.ForeignKey('lesson_template.id'),
        nullable=True
    ),
    sa.Column(
        'hall_id',
        None,
        sa.ForeignKey('hall.id')
    ),
    sa.Column(
        'booking_places_count',
        sa.Integer,
        sa.CheckConstraint('booking_places_count >= 0'),
        nullable=False
    ),
    sa.Column(
        'trial_places_count',
        sa.Integer,
        sa.CheckConstraint('trial_places_count >= 0'),
        server_default=sa.text('0'),
        nullable=False
    ),
    sa.Column(
        'coach_id',
        None,
        sa.ForeignKey('auth_user.id'),
        nullable=False
    ),
    ),

# Time when lesson starts
sa.Column(
    'time',
    sa.DateTime,
    nullable=False
),

# Lesson duration in minutes
sa.Column(
    'duration',

```

```

        sa.Integer,
        nullable=False,
        server_default=sa.text('60')
    ),
)

```

```

LessonVisitor = sa.Table(
    'lesson_visitor',
    metadata,
    sa.Column(
        'id',
        sa.Integer,
        primary_key=True,
        index=True
    ),
    sa.Column(
        'user_id',
        None,
        sa.ForeignKey('auth_user.id'),
        nullable=False
    ),
    sa.Column(
        'lesson_id',
        None,
        sa.ForeignKey('lesson.id'),
        nullable=False
    ),
    sa.Column(
        'lesson_purchase_id',
        None,
        sa.ForeignKey('lesson_purchase.id'),
        nullable=True
    ),
    sa.Column(
        'paid_on_lesson',
        sa.Boolean,
        server_default=sa.text('false'),
        nullable=False
    ),
    sa.Column(
        'paid_online',
        sa.Boolean,
        server_default=sa.text('false'),
        nullable=False
    ),
    sa.Column(
        'is_first_visit',
        sa.Boolean,
        server_default=sa.text('false'),
        nullable=False
    ),
)

```

Змн.	Арк.	№ докум.	Підпис	Дата

```

sa.Column(
    'pay_amount',
    sa.Integer,
    sa.CheckConstraint('pay_amount >= 0'),
    nullable=False
),
sa.Column(
    'moved_to_lesson_id',
    None,
    sa.ForeignKey('lesson.id'),
    nullable=True
),
# TODO (Inalkman): add freeze reference
sa.Column(
    'visited',
    sa.Boolean,
    server_default=sa.text('false'),
    nullable=False
)
)
)

```

```

LessonPurchase = sa.Table(
    'lesson_purchase',
    metadata,
    sa.Column(
        'id',
        sa.Integer,
        primary_key=True,
        index=True
    ),
    sa.Column(
        'user_id',
        None,
        sa.ForeignKey('auth_user.id'),
        nullable=False
    ),
    sa.Column(
        'lesson_subscription_id',
        None,
        sa.ForeignKey('lesson_subscription.id'),
        nullable=False,
    ),
    sa.Column(
        'freeze_used',
        sa.Boolean,
        server_default=sa.text('false'),
        nullable=False
    ),
    sa.Column(
        'available_movements',
        sa.Integer,

```

Змн.	Арк.	№ докум.	Підпис	Дата

```

server_default=sa.text('0'),
nullable=False
)
)

# docker-compose.yml
version: '3.2'

services:

  ballet_school_app: &app
  build:
    context: .
    args:
      requirements: requirements/development.txt
    container_name: ballet_school
  restart: always
  volumes:
    - ./app:delegated
  depends_on:
    - redis
  ports:
    - 8080:8080
    - 8081:8081
  command: adev runserver ./ballet_school/__main__.py -p 8080

  redis:
    image: redis:4
    ports:
      - 6379:6379

  test:
    <<: *app
    container_name: ballet_school_test
    ports:
      - 8082:8082
    command: make _test

  flake8-ci:
    build:
      context: .
      dockerfile: ./config/docker/Dockerfile-flake8
      args:
        FLAKE8_VERSION: 3.7.7
    volumes:
      - ./app:delegated
    command: .

  mypy-ci:
    build:

```

```

context: .
dockerfile: ./config/docker/Dockerfile-mypy
args:
  MYPY_VERSION: '0.710'
volumes:
  - ./app:delegated
command: --config-file mypy.ini -p ballet_school

```

```

tslint-ci:
build:
  context: .
  dockerfile: ./config/docker/Dockerfile-tslint
volumes:
  - ./ballet_school/js/js:delegated
command: -c tslint.json 'src/**/*.ts' 'src/**/*.tsx'

```

```

# Dockerfile
FROM python:3.7-slim

WORKDIR /app
ARG requirements=requirements/production.txt

```

```

ADD requirements /app/requirements

RUN apt-get update \
  && apt-get install -y gcc git python3-dev musl-dev libpq-dev \
  && python -m pip install --no-cache-dir -r $requirements \
  && apt-get remove -y gcc git \
  && apt-get -y autoremove

```

```

ADD . /app

```

```

# admin/views.py
import functools
import hashlib
import io

from aiohttp import web
from aiohttp.web_request import FileField
from aiohttp_security import check_permission

from ballet_school.constants import StorageBuckets
from ballet_school.utils.common import (
    landing_media_url
)

```

```

class FileUpload(web.View):

```

```

    model_bucket = {

```

					ДП ІС-6211.00.002 ПЗ	Арк.
						82
Змн.	Арк.	№ докум.	Підпис	Дата		

```
'page': StorageBuckets.LANDING
}

@staticmethod
def get_hashed_filename(file_field: FileField) -> str:
    file = file_field.file
    original_name = file_field.filename
    hash_ = hashlib.md5()

    # 1MB
    read_chunk = functools.partial(file.read, 1024 * 1024)
    chunks = iter(read_chunk, b'')

    for chunk in chunks:
        hash_.update(chunk)
    file.seek(0)

    hash_hex_digest = hash_.hexdigest()
    file_type = original_name[original_name.index('.'): ]

    return f'{hash_hex_digest}{file_type}'

@staticmethod
def get_storage_key(
    model_name: str,
    primary_key: str,
    field: str,
    filename: str,
) -> str:
    return f'{model_name}/{primary_key}/{field}/{filename}'

async def upload_file(self, file: io.BytesIO, key: str) -> None:
    storage = self.request.config_dict['storage']

    await storage.put_object(
        ACL='public-read',
        Bucket=StorageBuckets.LANDING,
        Key=key,
        Body=file.read()
    )

async def post(self) -> web.Response:
    await check_permission(self.request, 'admin')
    match_info = self.request.match_info

    model_name = match_info['model_name']
    primary_key = match_info['primary_key']
    field = match_info['field']
    bucket = self.model_bucket.get(model_name)

    if bucket is None:
        return web.Response(
```

Змн.	Арк.	№ докум.	Підпис	Дата

```
body=f'Can't find bucket for model {model_name}',
status=404
)

data = await self.request.post()
file_field = data.get('file')

upload_key = self.get_storage_key(
    model_name=model_name,
    primary_key=primary_key,
    field=field,
    filename=self.get_hashed_filename(
        file_field=file_field
    )
)
await self.upload_file(
    file=file_field.file,
    key=upload_key
)

return web.json_response({
    'url': landing_media_url(upload_key),
})
```

```
# spa_entrpoint.html
<html lang="ru">
<head>
<title>Title here</title>
<meta charset="UTF-8">
<meta
    name="viewport"
    content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0"
>
<meta
    http-equiv="X-UA-Compatible"
    content="ie=edge"
>
<link
    rel="stylesheet"
    href="{ { css_bundle(bundle_name) } }"
>
<style>
@font-face {
    font-family: "Gilroy";
    font-weight: normal;
    font-style: normal;
    src: url({ { static('fonts/Gilroy/GilroyRegular/GilroyRegular.eot') } });
    src: local("☐"), url({ { static('fonts/Gilroy/GilroyRegular/GilroyRegular.eot?#iefix') } }) format("embedded-opentype"),
url({ { static('fonts/Gilroy/GilroyRegular/GilroyRegular.woff') } }) format("woff"), url({ { static('fonts/Gilroy/GilroyRegular/GilroyRegular.ttf')
}}) format("truetype");
```

```
}

@font-face {
  font-family: "Gilroy";
  font-weight: 500;
  font-style: normal;
  src: url({{ static('fonts/Gilroy/GilroyMedium/GilroyMedium.eot') }});
  src: local("□ □"), url({{ static('fonts/Gilroy/GilroyMedium/GilroyMedium.eot?#iefix') }}) format("embedded-opentype"),
url({{ static('fonts/Gilroy/GilroyMedium/GilroyMedium.woff') }}) format("woff"), url({{ static('fonts/Gilroy/GilroyMedium/GilroyMedium.ttf')
}}) format("truetype");
}

@font-face {
  font-family: "Gilroy";
  font-weight: 700;
  font-style: normal;
  src: url({{ static('fonts/Gilroy/GilroyBold/GilroyBold.eot') }});
  src: url({{ static('fonts/Gilroy/GilroyBold/GilroyBold.eot?#iefix') }}) format("embedded-opentype"), url({{
static('fonts/Gilroy/GilroyBold/GilroyBold.woff') }}) format("woff"), url({{ static('fonts/Gilroy/GilroyBold/GilroyBold.ttf') }})
format("truetype");
}

body {
  font-family: Gilroy, serif;
}

</style>
<link rel="apple-touch-icon" sizes="180x180" href="{{ static('apple-touch-icon.png') }}">
<link rel="icon" type="image/png" sizes="32x32" href="{{ static('favicon-32x32.png') }}">
<link rel="icon" type="image/png" sizes="16x16" href="{{ static('favicon-16x16.png') }}">
<link rel="manifest" href="{{ static('site.webmanifest') }}">
<link rel="mask-icon" href="{{ static('safari-pinned-tab.svg') }}" color="#5bbad5">
<meta name="msapplication-TileColor" content="#da532c">
<meta name="theme-color" content="#ffffff">

</head>
<body style="padding: 0;margin: 0;">
  <div id="app"></div>

  <script src="{{ js_bundle(bundle_name) }}"></script>
</body>
</html>
```

```
# migrations/env.py
from logging.config import fileConfig

from alembic import context
from alembic.util import CommandError
from sqlalchemy import (
    engine_from_config,
    pool
)

from ballet_school import migrations

# import here you table files for autodetect your tables by the alembic
```

					ДП ІС-6211.00.002 ПЗ	Арк.
						85
Змн.	Арк.	№ докум.	Підпис	Дата		

```

from ballet_school.apps.auth.tables import * # noqa
from ballet_school.apps.school.tables import * # noqa
from ballet_school.apps.main.tables import * # noqa

# this is the Alembic Config object, which provides
# access to the values within the .ini file in use.
config = context.config

# Interpret the config file for Python logging.
# This line sets up loggers basically.
fileConfig(config.config_file_name)

# add your model's MetaData object here
# for 'autogenerate' support
# from myapp import mymodel
# target_metadata = mymodel.Base.metadata
target_metadata = migrations.metadata

# other values from the config, defined by the needs of env.py,
# can be acquired:
# my_important_option = config.get_main_option("my_important_option")
# ... etc.

def run_migrations_offline() -> None:
    """Run migrations in 'offline' mode.

    This configures the context with just a URL
    and not an Engine, though an Engine is acceptable
    here as well.  By skipping the Engine creation
    we don't even need a DBAPI to be available.

    Calls to context.execute() here emit the given string to the
    script output.

    """
    url = config.get_main_option("sqlalchemy.url")
    context.configure(
        url=url,
        target_metadata=target_metadata,
        literal_binds=True,
        compare_type=True
    )

    with context.begin_transaction():
        context.run_migrations()

def run_migrations_online() -> None:
    """Run migrations in 'online' mode.

```

Змн.	Арк.	№ докум.	Підпис	Дата

In this scenario we need to create an Engine
and associate a connection with the context.

```
"""
```

```
connectable = engine_from_config(
    config.get_section(config.config_ini_section),
    prefix='sqlalchemy.',
    poolclass=pool.NullPool)
```

```
with connectable.connect() as connection:
```

```
    context.configure(
        connection=connection,
        target_metadata=target_metadata,
        compare_type=True
    )
```

```
with context.begin_transaction():
```

```
    try:
        context.run_migrations()
    except CommandError as e:
        if "Can't locate revision identified by" in str(e):
            print(f'Location error in migration {e}')
        else:
            raise e
```

```
if context.is_offline_mode():
```

```
    run_migrations_offline()
```

```
else:
```

```
    run_migrations_online()
```

```
# alembic.ini
```

```
# about default section watch here
```

```
# https://www.enfoldsystems.com/software/proxy/docs/4.0/configuringmanually.html#the-default-section
```

```
[DEFAULT]
```

```
# path to migration scripts
```

```
script_location = ballet_school/migrations
```

```
# template used to generate migration files
```

```
# file_template = %(rev)s_%(slug)s
```

```
# timezone to use when rendering the date
```

```
# within the migration file as well as the filename.
```

```
# string value is passed to dateutil.tz.gettz()
```

```
# leave blank for localtime
```

```
# timezone =
```

```
# max length of characters to apply to the
```

```
# "slug" field
```

					ДП ІС-6211.00.002 ПЗ	Арк.
						87
Змн.	Арк.	№ докум.	Підпис	Дата		

```
#truncate_slug_length = 40

# set to 'true' to run the environment during
# the 'revision' command, regardless of autogenerate
# revision_environment = false

# set to 'true' to allow .pyc and .pyo files without
# a source .py file to be detected as revisions in the
# versions/ directory
# sourceless = false

# version location specification; this defaults
# to alembic/versions. When using multiple version
# directories, initial revisions must be specified with --version-path
# version_locations = %(here)s/bar %(here)s/bat alembic/versions

# the output encoding used when revision files
# are written from script.py.mako
# output_encoding = utf-8

# Logging configuration
[loggers]
keys = root,sqlalchemy,alembic

[handlers]
keys = console

[formatters]
keys = generic

[logger_root]
level = WARN
handlers = console
qualname =

[logger_sqlalchemy]
level = WARN
handlers =
qualname = sqlalchemy.engine

[logger_alembic]
level = INFO
handlers =
qualname = alembic

[handler_console]
class = StreamHandler
args = (sys.stderr,)
level = NOTSET
formatter = generic

[formatter_generic]
```

```
format = %(levelname)-5.5s [% (name)s] %(message)s
```

```
datefmt = %H:%M:%S
```

```
# Makefile
```

```
PROJECT_NAME=ballet_school
```

```
# Common
```

```
all: run
```

```
run:
```

```
@docker-compose up ballet_school_app
```

```
stop:
```

```
@docker-compose stop
```

```
clean:
```

```
@docker-compose down
```

```
bash:
```

```
@docker exec -it ballet_school bash
```

```
# Linters & tests
```

```
mypy:
```

```
@docker-compose run --rm $(PROJECT_NAME)_app mypy $(PROJECT_NAME)
```

```
lint:
```

```
@docker-compose run --rm $(PROJECT_NAME)_app flake8 $(PROJECT_NAME)
```

```
_test:
```

```
# todo: remove no:warnings
```

```
@py.test -p no:warnings --cov
```

```
test: lint
```

```
@docker-compose up test
```

```
@docker-compose stop test
```

```
# Database
```

```
psql:
```

```
@docker exec -it ballet_school_postgres psql -U postgres
```

```
migrations:
```

```
@docker exec -it ballet_school_app alembic -n alembic:dev revision --autogenerate;
```

```
migrate:
```

```
@docker exec -it ballet_school_app alembic -n alembic:dev upgrade head;
```

```
adev: wait_resources
```

```
adev runserver ./ballet_school/__main__.py -p 8080
```

Змн.	Арк.	№ докум.	Підпис	Дата

wait_resources:

python3 -m ballet_school.utils.wait_script

lesson_visitor/mutations.py

import graphene

from graphql import ResolveInfo

from sqlalchemy import not_

from sqlalchemy.sql import Select

from ballet_school.apps.auth.utils import get_user

from ballet_school.apps.school.db_utils import is_user_coach_for_lesson_visitor

from ballet_school.apps.school.tables import LessonVisitor

from ballet_school.graph.enums import ErrorEnum

from ballet_school.graph.objects.error import Error

from ballet_school.graph.utils import format_single_field_error

class TogglePayment(graphene.Mutation):

class Arguments:

visitor_id = graphene.Int(

required=True

)

lesson_visitor = graphene.Field(

'ballet_school.graph.objects.lesson_visitor.LessonVisitor'

)

errors = graphene.List(

Error

)

@staticmethod

def get_toggle_payment_query(visitor_id: int) -> Select:

return (

LessonVisitor

.update()

.values({

LessonVisitor.c.paid_on_lesson:

not_(LessonVisitor.c.paid_on_lesson)

}))

.where(LessonVisitor.c.id == visitor_id)

.returning(LessonVisitor)

)

async def mutate(

self,

info: ResolveInfo,

visitor_id: int

) -> 'TogglePayment':

request = info.context['request']

					ДП ІС-6211.00.002 ПЗ	Арк.
						90
Змн.	Арк.	№ докум.	Підпис	Дата		

```

app = request.app
user = await get_user(request)

if not user.is_coach:
    return TogglePayment(
        errors=[format_single_field_error(
            code=ErrorEnum.NOT_COACH,
            message='Даное действие доступно только тренеру'
        )]
    )

async with app['db'].acquire() as conn:
    is_valid_coach = await is_user_coach_for_lesson_visitor(
        conn=conn,
        coach_id=user.id,
        lesson_visitor_id=visitor_id
    )

    if not is_valid_coach:
        return TogglePayment(
            errors=[format_single_field_error(
                code=ErrorEnum.NOT_COACH,
                message=(
                    'Изменить статус оплаты может только тренер'
                    'этого посетителя'
                )
            )]
        )

    query = TogglePayment.get_toggle_payment_query(
        visitor_id=visitor_id
    )

    rows = await conn.execute(query)
    visitor = await rows.fetchone()

    return TogglePayment(
        lesson_visitor=visitor
    )

```

```

class ToggleVisit(graphene.Mutation):

```

```

    class Arguments:

```

```

        visitor_id = graphene.Int(
            required=True
        )

```

```

        lesson_visitor = graphene.Field(
            'ballet_school.graph.objects.lesson_visitor.LessonVisitor'
        )

```

					ДП ІС-6211.00.002 ПЗ	Арк.
						91
Змн.	Арк.	№ докум.	Підпис	Дата		

```

errors = graphene.List(
    Error
)

@staticmethod
def get_toggle_visit_query(visitor_id: int) -> Select:
    return (
        LessonVisitor
        .update()
        .values({
            LessonVisitor.c.visited:
                not_(LessonVisitor.c.visited)
        })
        .where(LessonVisitor.c.id == visitor_id)
        .returning(LessonVisitor)
    )

async def mutate(
    self,
    info: ResolveInfo,
    visitor_id: int
) -> 'TogglePayment':
    request = info.context['request']
    app = request.app
    user = await get_user(request)

    if not user.is_coach:
        return TogglePayment(
            errors=[format_single_field_error(
                code=ErrorEnum.NOT_COACH,
                message='Даное действие доступно только тренеру'
            )]
        )

    async with app['db'].acquire() as conn:
        is_valid_coach = await is_user_coach_for_lesson_visitor(
            conn=conn,
            coach_id=user.id,
            lesson_visitor_id=visitor_id
        )

        if not is_valid_coach:
            return TogglePayment(
                errors=[format_single_field_error(
                    code=ErrorEnum.NOT_COACH,
                    message=(
                        'Изменить статус присутствия может только тренер'
                        'этого посетителя'
                    )
                )]
            )

```

Змн.	Арк.	№ докум.	Підпис	Дата

```
        query = ToggleVisit.get_toggle_visit_query(
            visitor_id=visitor_id
        )

        rows = await conn.execute(query)
        visitor = await rows.fetchone()

    return TogglePayment(
        lesson_visitor=visitor
    )

# user/mutations.py
from typing import (
    List,
    Optional,
    NamedTuple
)

import graphene
from aiohttp import ClientSession
from aiopg.sa import SAConnection
from graphql import ResolveInfo
from aiohttp_session import get_session
from sqlalchemy import (
    select,
    and_
)

from ballet_school.apps.auth.db_utils import (
    create_user,
    is_phone_number_in_use,
    create_social_auth
)

from ballet_school.apps.auth.tables import (
    User as UserTable,
    UserSocialAuth
)

from ballet_school.graph.models.user import User
from ballet_school.graph.objects.error import (
    Error,
    FieldError
)

from ballet_school.graph.enums import (
    ErrorEnum,
    SocialAuthProviderEnum
)

from ballet_school.apps.auth.enums import (
    SocialAuthProviderEnum as PySocialAuthProviderEnum
)

from ballet_school.apps.auth.utils import (
    authenticate,
```

					ДП ІС-6211.00.002 ПЗ	Арк.
						93
Змн.	Арк.	№ докум.	Підпис	Дата		

```

generate_user_password,
normalize_phone_number,
get_google_user_profile,
get_facebook_user_profile,
_force_authenticate,
SocialUserProfile
)
from ballet_school.apps.auth.exceptions import InvalidCredentialsError
from ballet_school.constants import SessionKeys

class PhoneVerificationResult(NamedTuple):
    phone_number: str = None
    errors: List[Error] = None

class AuthenticateUser(graphene.Mutation):

    class Arguments:
        login = graphene.String()
        password = graphene.String()

    is_success = graphene.Boolean(
        required=True
    )
    errors = graphene.List(
        Error
    )

    async def mutate(
        self,
        info: ResolveInfo,
        login: str,
        password: str
    ) -> 'AuthenticateUser':
        request = info.context['request']

        try:
            await authenticate(
                request=request,
                phone_number=login,
                password=password
            )
        except InvalidCredentialsError:
            return AuthenticateUser(
                is_success=False,
                errors=[
                    Error(errors=[
                        FieldError(
                            code=ErrorEnum.INVALID_CREDENTIALS,
                            message='Номер телефона или пароль указаны неверно'
                        )
                    ])
                ]
            )

```

```

    })
    ]
    )
    return AuthenticateUser(is_success=True)

```

```
class SendPasswordSMS(graphene.Mutation):
```

```
class Arguments:
```

```
    phone_number = graphene.String()
```

```
is_success = graphene.Boolean(
```

```
    required=True
```

```
)
```

```
errors = graphene.List(
```

```
    Error
```

```
)
```

```
@staticmethod
```

```
def format_sms_text(password: str) -> str:
```

```
    return (
```

```
        'Ваш пароль:\n' +
```

```
        ''.join([password[:3], password[3:6], password[6:9]])
```

```
)
```

```
@staticmethod
```

```
async def validate_phone_number(
```

```
    conn: SAConnection,
```

```
    phone_number: str
```

```
) -> PhoneVerificationResult:
```

```
    """
```

```
    Validate phone_number, check format and whether it is not used by
    another user.
```

```
    """
```

```
    errors: List[Error] = []
```

```
    normalized_phone_number = normalize_phone_number(phone_number)
```

```
    if normalized_phone_number is None:
```

```
        errors.append(
```

```
            Error(
```

```
                field='phoneNumber',
```

```
                errors=[
```

```
                    FieldError(
```

```
                        code=ErrorEnum.INVALID_PHONE_NUMBER,
```

```
                        message='Номер телефона указан неверно'
```

```
                    ),
```

```
                ]
```

```
            )
```

```
        )
```

```
    return PhoneVerificationResult(errors=errors)
```

```
    phone_already_in_use = await is_phone_number_in_use(
```

					ДП ІС-6211.00.002 ПЗ	Арк.
						95
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        conn=conn,
        phone_number=normalized_phone_number
    )
    if phone_already_in_use:
        errors.append(
            Error(
                field='phoneNumber',
                errors=[
                    FieldError(
                        code=ErrorEnum.PHONE_NUMBER_ALREADY_IN_USE,
                        message='Номер телефона уже используется'
                    ),
                ]
            )
        )
    return PhoneVerificationResult(errors=errors)

return PhoneVerificationResult(phone_number=normalized_phone_number)

async def mutate(
    self,
    info: ResolveInfo,
    phone_number: str
) -> 'SendPasswordSMS':
    request = info.context['request']
    db = request.app['db']
    esputnik = request.app['esputnik']

    async with db.acquire() as conn:
        normalized_phone_number, errors = (
            await SendPasswordSMS.validate_phone_number(
                conn=conn,
                phone_number=phone_number
            )
        )

    if errors:
        return SendPasswordSMS(is_success=False, errors=errors)

    session = await get_session(request)

    random_password = generate_user_password()
    sms_text = SendPasswordSMS.format_sms_text(random_password)
    await esputnik.send_sms(
        text=sms_text,
        phone_numbers=[normalized_phone_number]
    )
    session[SessionKeys.SMS_PASSWORD_PHONE] = normalized_phone_number
    session[SessionKeys.SMS_PASSWORD] = random_password

    return SendPasswordSMS(is_success=True)

```

Змн.	Арк.	№ докум.	Підпис	Дата

```

class VerifyNumber(graphene.Mutation):

    class Arguments:
        phone_number = graphene.String(required=True)
        password = graphene.String(required=True)

    is_success = graphene.Boolean(
        required=True
    )
    errors = graphene.List(
        Error
    )

    @staticmethod
    def validate(
        normalized_phone_number: str,
        password: str,
        session_phone_number: str,
        session_password: str
    ) -> List[Error]:
        errors: List[Error] = []
        if normalized_phone_number != session_phone_number:
            errors.append(
                Error(
                    field='phoneNumber',
                    errors=[
                        FieldError(
                            code=ErrorEnum.INVALID_PHONE_NUMBER,
                            message='Номер телефона указан неверно'
                        ),
                    ]
                )
            )

        if password != session_password:
            errors.append(
                Error(
                    field='password',
                    errors=[
                        FieldError(
                            code=ErrorEnum.INVALID_PASSWORD,
                            message='Код указан неверно'
                        ),
                    ]
                )
            )

        return errors

    @staticmethod
    async def mutate(

```

Змн.	Арк.	№ докум.	Підпис	Дата

```

    info: ResolveInfo,
    phone_number: str,
    sms_password: str
) -> 'VerifyNumber':
    request = info.context['request']
    session = await get_session(request)

    normalized_phone_number = normalize_phone_number(phone_number)
    session_phone_number = session.get(SessionKeys.SMS_PASSWORD_PHONE)
    session_password = session.get(SessionKeys.SMS_PASSWORD)

    if normalized_phone_number is not None:
        errors = VerifyNumber.validate(
            normalized_phone_number=normalized_phone_number,
            password=sms_password,
            session_phone_number=session_phone_number,
            session_password=session_password
        )
    else:
        errors = [Error(
            errors=[
                FieldError(
                    code=ErrorEnum.INVALID_FORMAT,
                    message='Неверный формат номера телефона'
                )
            ]
        )]

    if errors:
        return VerifyNumber(is_success=False, errors=errors)

    return VerifyNumber(is_success=True)

```

```
class RegisterUser(graphene.Mutation):
```

```
class Arguments:
```

```

    phone_number = graphene.String(required=True)
    password = graphene.String(required=True)
    first_name = graphene.String()
    last_name = graphene.String()
    access_token = graphene.String(
        description='Social networks access token'
    )
    social_type = graphene.Argument(
        SocialAuthProviderEnum,
        description='Social network type'
    )

```

```

    user = graphene.Field(User)
    errors = graphene.List(Error)

```

					ДП ІС-6211.00.002 ПЗ	Арк.
						98
Змн.	Арк.	№ докум.	Підпис	Дата		

```

@staticmethod
async def validate_token(
    client: ClientSession,
    provider: SocialAuthProviderEnum,
    access_token: str
) -> Optional[SocialUserProfile]:
    """
    Make request to social network to check token, returns social user info
    """
    user_data = None
    if provider == SocialAuthProviderEnum.GOOGLE:
        user_data = await get_google_user_profile(
            client=client,
            access_token=access_token
        )
    elif provider == SocialAuthProviderEnum.FACEBOOK:
        user_data = await get_facebook_user_profile(
            client=client,
            access_token=access_token
        )

    return user_data

@staticmethod
async def mutate(
    info: ResolveInfo,
    phone_number: str,
    password: str,
    first_name: str = None,
    last_name: str = None,
    access_token: str = None,
    social_type: SocialAuthProviderEnum = None
) -> 'RegisterUser':
    request = info.context['request']
    app = request.app

    async with app['db'].acquire() as conn:
        normalized_phone_number, errors = (
            await SendPasswordSMS.validate_phone_number(
                conn=conn,
                phone_number=phone_number
            )
        )

    if errors:
        return RegisterUser(errors=errors)

    phone_verification = await VerifyNumber.mutate(
        info=info,

```

Змн.	Арк.	№ докум.	Підпис	Дата

```

    phone_number=normalized_phone_number,
    sms_password=password
)

if not phone_verification.is_success:
    return RegisterUser(errors=phone_verification.errors)

social_user = None
if access_token and social_type:
    social_user = await RegisterUser.validate_token(
        client=app['client'],
        provider=SocialAuthProviderEnum.get(social_type),
        access_token=access_token
    )

# TODO(Inalkman): password encode errors
async with app['db'].acquire() as conn:
    rows = await create_user(
        conn=conn,
        phone_number=normalized_phone_number,
        encoded_password=password.encode('utf-8'),
        first_name=(
            social_user.first_name
            if social_user is not None
            else first_name
        ),
        last_name=(
            social_user.last_name
            if social_user is not None
            else last_name
        ),
        email=(
            social_user.email
            if social_user is not None
            else None
        ),
        avatar_url=(
            social_user.picture_url
            if social_user is not None
            else None
        ),
        returning=(
            UserTable.c.id,
            UserTable.c.first_name,
            UserTable.c.last_name,
            UserTable.c.phone_number
        )
    )
    user = await rows.first()
    if social_user is not None:
        await create_social_auth(
            conn=conn,

```

					ДП ІС-6211.00.002 ПЗ	Арк.
						10
Змн.	Арк.	№ докум.	Підпис	Дата		н

```

        user_id=user.id,
        uid=str(social_user.id),
        provider=PySocialAuthProviderEnum(social_type)
    )
    await _force_authenticate(
        request=request,
        user_id=user.id
    )

    return RegisterUser(
        user=User(**user)
    )

class SocialAuthenticateUser(graphene.Mutation):

    class Arguments:
        social_type = graphene.Argument(
            SocialAuthProviderEnum,
            description='Social network type',
            required=True
        )
        access_token = graphene.String()

    registration_required = graphene.Boolean(
        description='Token valid, but user not found, registration required'
    )
    is_success = graphene.Boolean(required=True)

    @staticmethod
    async def mutate(
        _,
        info: ResolveInfo,
        social_type: str,
        access_token: str
    ) -> 'SocialAuthenticateUser':
        request = info.context['request']
        app = request.app
        provider = SocialAuthProviderEnum.get(social_type)

        social_user = await RegisterUser.validate_token(
            client=app['client'],
            provider=provider,
            access_token=access_token
        )

        if social_user is None:
            return SocialAuthenticateUser(is_success=False)

        async with app['db'].acquire() as conn:
            social_auth_condition = and_(
                UserSocialAuth.c.uid == social_user.id,

```

Змн.	Арк.	№ докум.	Підпис	Дата

```
UserSocialAuth.c.provider ==
    SocialAuthProviderEnum.get(social_type)
)
rows = await conn.execute(
    select([UserSocialAuth.c.user_id])
    .where(social_auth_condition)
)
social_auth = await rows.fetchone()

if social_auth:
    await _force_authenticate(
        request=request,
        user_id=social_auth.user_id
    )

    return SocialAuthenticateUser(is_success=True)

return SocialAuthenticateUser(
    is_success=False,
    registration_required=True
)
```

					ДП ІС-6211.00.002 ПЗ	Арк.
						10
Змн.	Арк.	№ докум.	Підпис	Дата		2