Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Свириденка Ігоря Євгеновича*
(ПІБ)

академічної групи *121-17-1*
(шифр)

спеціальності *121 Інженерія програмного забезпечення*
(код і назва спеціальності)

освітньої програми *Інженерія програмного забезпечення*
(назва освітньої програми)

на тему: *Розробка серверної частини соціальної мережі для студентів*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	*доц. Реута О. В.*			
розділів:				
спеціальний	*доц. Реута О. В.*			
економічний	*доц. Касьяненко Л.В.*			
Рецензент				
Нормоконтролер	*доц. Гуліна І.Г.*			

Дніпро
2021

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

_____ І.М. Удовик
(підпис) (прізвище, ініціали)

« » _____ 2021 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *121-17-1* (група) *Свириденка Ігоря Євгеновича* (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка серверної частини соціальної мережі для студентів*

затверджена наказом ректора НТУ «ДП» від *07.06.2021р.* № *317-с*

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів виробничої практики та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі.*** ***Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***13.05.2021 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПО й тривалості його розробки***	***27.05.2021 р.***

Завдання видав _____ *доц. Реута О.В.*
(підпис) (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ *Свириденко І. Є.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14.01.2021 р.*

Термін подання кваліфікаційної роботи до ЕК: *10.06.2021 р.*

РЕФЕРАТ

Пояснювальна записка: 53 с., 32 рис., 6 табл., 3 дод., * джерела.

Об'єкт розробки: соціальна мережа для студентів та викладачів

Мета кваліфікаційної роботи: розробка універсальної серверної частини соціальної мережі для студентів та викладачів, яка надаватиме їм можливість встановлювати зв'язок між собою, створювати групи, інформаційні оголошення, встановлювати та дивитися розклад занять, екзаменів та учбового процесу, комунікувати за допомогою чату.

У вступі виконується аналіз сучасного стану проблеми, уточняється постановка завдання, мета кваліфікаційної роботи та галузь її застосування, обґрунтовується актуальність теми.

У першому розділі проводиться дослідження предметної області та існуючих рішень, визначається актуальність завдання та призначення розробки, розроблюється постановка завдання.

У другому розділі обирається платформа для розробки, виконується проектування програми і її розробка, наводиться опис структури функціонування системи, описується робота програми.

В економічному розділі визначається трудомісткість розробленого програмного продукту, проводиться підрахунок вартості роботи по створенню застосунку та розраховується час на його створення.

Практичне значення полягає в розробці серверної частини веб-додатку соціальна мережа для студентів, що полегшить взаємодію між студентами та між викладачами та допоможе їм будувати ефективну соціальну комунікацію.

Актуальність програмного продукту визначається малою кількістю подібних публічних продуктів, орієнтовану на широку громаду. Кожний вищий навчальний заклад має свою соціальну екосистему в інтернеті. Розроблюваний додаток допоможе будувати плідну комунікацію між студентами та викладачами незалежно від їхнього місця навчання, адже він намагається увібрати в себе загальні найважливіші функції, які необхідні як студентам, так і викладачам

Список ключових слів: ПРОГРАМА, УНІВЕРСИТЕТ, МЕРЕЖА, САЙТ, БРАУЗЕР, СЕРВЕР, ІНФОРМАЦІЙНА СИСТЕМА, ЕОМ, ПРИСТРІЙ.

ABSTRACT

Explanatory note: 53 p., 32 figs., 6 tabl., 3 appx., 22 sources.

Object of development: social network for students and teachers

The purpose of the qualification work: development of a general-purpose server-side API of the social network for students and teachers, which will be provided with functionality to connect with each other, create groups, posts, set and view schedules for groups, examine learning process etc.

The introduction analyzes the current state of the problem, clarifies the problem, the purpose of the qualification work and the scope of its application, substantiates the relevance of the topic.

In the first section the research of the subject area and existing decisions is carried out, the urgency of the task and purpose of development is defined, the statement of the task is developed.

In the second section, the platform for development is selected, the program is designed and developed, a description of the structure of the system, describes the operation of the program.

In the economic section, the complexity of the developed software product is determined, the cost of work on creating the application is calculated and the time for its creation is calculated.

Of practical importance is the development of the server part of the web application social network for students, which will facilitate interaction between students and between teachers and help them build effective social communication.

The relevance of the software product is determined by the small number of such public products, targeted at the general community. Every higher education institution has its own social ecosystem on the Internet. The developed application will help to build fruitful communication between students and teachers, regardless of their place of study, because it tries to absorb the most important common functions that are needed by both students and teachers.

Keywords: PROGRAM, UNIVERSITY, NETWORK, SITE, BROWSER, SERVER, INFORMATION SYSTEM, COMPUTER, DEVICE.

ЗМІСТ

СПИСОК УМОВНИХ ПОЗНАЧЕНЬ

API – інтерфейс для програмування застосунків;

БД – база даних;

ООП – об'єктно-орієнтоване програмування;

ОС – операційна система;

ПК – персональний комп’ютер;

ПЗ – програмне забезпечення;

ІТ – інформаційні технології.

ВСТУП

В наш час соціальні мережі широко розповсюджені серед людей різного віку. Поняття «соціальна мережа» може бути визначена як соціальна структура, утворена індивідами або організаціями. Вона відображає розмаїті зв'язки між ними через різноманітні соціальні взаємовідносини, починаючи з випадкових знайомств і закінчуючи тісними родинними зв'язками. У наш час в Україні вчиться понад 1,3 мільйонів студентів у вищих навчальних закладах на території всієї України, котрим надзвичайно важливо максимально ефективно комунікувати між собою та викладачами. Іноді це відбувається за допомогою різноманітних месенджерів, таких як Viber, Telegram або WhatsApp, іноді за допомогою програм для відео конференцій, наприклад: Zoom, Microsoft Teams, Google Meets. Наявність стількох багатьох способів комунікації з одного боку говорить про обізнаність та відповідальність викладачів, а з другого – про відсутність централізованої соціальної екосистеми, спрямованої на взаємодію між студентами та викладачами.

Отже, завданням даного дипломного проекту є створення універсальної серверної частини соціальної адреси для студентів та викладачів.

Завдання даної кваліфікаційної роботи та об'єкт його діяльності безпосередньо пов'язані з напрямом підготовки «Інженерія програмного забезпечення» та відповідає узагальненій тематиці кваліфікаційних робіт і переліку зазначених виробничих функцій, типових задач діяльності, умінню та компетенціям, якими повинні володіти бакалаври напряму 121 «Інженерія програмного забезпечення» галузі знань 12 «Інформаційні технології».

Завдання даного дипломного проекту та об'єкт його діяльності безпосередньо пов'язані з напрямом підготовки та відповідає узагальненій тематиці кваліфікаційних робіт і переліку зазначених виробничих функцій, типових задач діяльності, умінню та компетенціям, якими повинні володіти бакалаври напряму 121 «Інженерія програмного забезпечення».

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Загальні відомості з предметної галузі

На сучасному етапі світового розвитку набуває нового значення мережа Інтернет і все, що з нею пов‘язано. Людство у своєму розвитку наблизилось до того, що Інтернет став основним генератором світових макротенденцій. Він сприяє трансформації ціннісних орієнтирів людства та його соціальних структур. Останнім часом ми не можемо уявити свого життя без Інтернету[5]. Він міцно увійшов в наше повсякденне життя. Ми активно користуємося Інтернетом вдома, на роботі, з розвитком нових технологій Інтернет перекочував в наші мобільні телефони і смартфони. Що дало нам можливість практично весь час перебувати онлайн. Поява Інтернету радикально змінила форми, зміст, механізми, функції соціальних комунікацій. З розвитком Інтернету, з'явилася можливість використовувати всі його досягнення в різних його проявах. Одним з таких проявів стали соціальні мережі, які набули на сьогодні статусу невід'ємного атрибуту нашого життя. Представити сучасну людину без соціальних мереж просто неможливо. Спілкування, пошук інформації і друзів, обмін новинами, можливість слухати музику, дивитися відео і фотографії. Складно собі уявити, що колись люди могли обійтися без профілю в соціальній мереж[6].

Якщо звернутись до вчень видатного американського психолога Абрахама Маслоу, то сьогоднішня необхідність у соціальних мережах може бути обумовлена соціальної сходинкою у представленій піраміді (рис. 1.1): кожна людина хоче бути соціально-значимою в суспільстві, бути почутою. Також, крім соціальної сходинки, подібні мережі можуть бути використані як способи самовираження та самореалізації: у наш час багато людей, котрі зареєстровані на подібних ресурсах, викладають на свої сторінки багато медіа, де показують свої таланти.

Рис. 1.1 Піраміда Маслоу

Взаємозалежність може базуватися на:

– Дружба
– Спорідненість
– Спільні інтереси
– Фінансовий та професійний обмін

Люди формують дружні стосунки природним шляхом. Ці з’єднання співіснують в офлайн-світі та в Інтернеті. Поняття «дружба» розуміє під собою близьку прив'язанність. Дружба відзначається спільністю цілей та інтересів, міцністю та тривалістю зв'язків, взаємною привабливістю, довірою та відданістю[9].

Іноді люди збираються довкола спільних інтересів**,** таких як мистецтво, музика чи книги. Об’єктом інтересу є те, що об’єднує громаду. Соціальні мережі допомагають цим громадам залишатися на зв’язку та розвивати спільноти у всьому світі.

Фінансовий та професійний обмін також може прокласти шлях до соціальних взаємодій. Щоденно сотні людей обмінюються накопиченим досвідом та навіть можуть знайти роботу у відповідних для цього соціальних мережах: LinkedIn, Spiceworks або ж Doximity.

За доступністю можна виділити такі типи соціальних мереж: закриті; відкриті; змішані. Нині більшість соціальних мереж повністю відкриті для усіх, хоча деякі проекти із-за своєї бізнес-моделі не передбачають публічності, і тому вони вже із самого початку створювалися закритими. Соціальні. мережі змішаного типу розвиваються дуже погано: основним завданням у них є досягнення популярності такого рівня, як і у відкритих мереж, але користувачам не подобаються різні бар'єри і тому вони досить неохоче приєднуються до таких соціальних мереж.

За спрямуванням соціальні мережі можна розділити на:

1. особисті
2. професійні
3. тематичні

Особисті спрямовані на підтримку і налагодження вже існуючих контактів, а також для пошуку нових. Професійні спрямовані на професійний розвиток та побудову кар’єри. Тематичні збирають аудиторію за певними інтересами: музика, хобі тощо[10].

Для виявлення переваг й недоліків існуючих соціальних мереж, що схожі на розроблюваний додаток, варто розглянути такі приклади, як **Scientific Social Community**, **Brainly та Facebook**.

Scientific Social Community

Scientific Social Community була створена у 2008 р. студентами-ентузіастами за допомогою підтримки благодійного фонду.

Вона надає можливості для пошуку фінансування грантів й роботи за кордоном, а також для знаходження колег в різних галузях наукових інтересів. Scientific Social Community містить спеціалізовані інструменти, спрямовані саме на вирішення проблем пошуку інформації вченими. Варто відзначити, що надані даним проектом сервіси представляють інтерес як для заслужених і відомих вчених, так і для студентів і аспірантів, які активно займаються наукою.

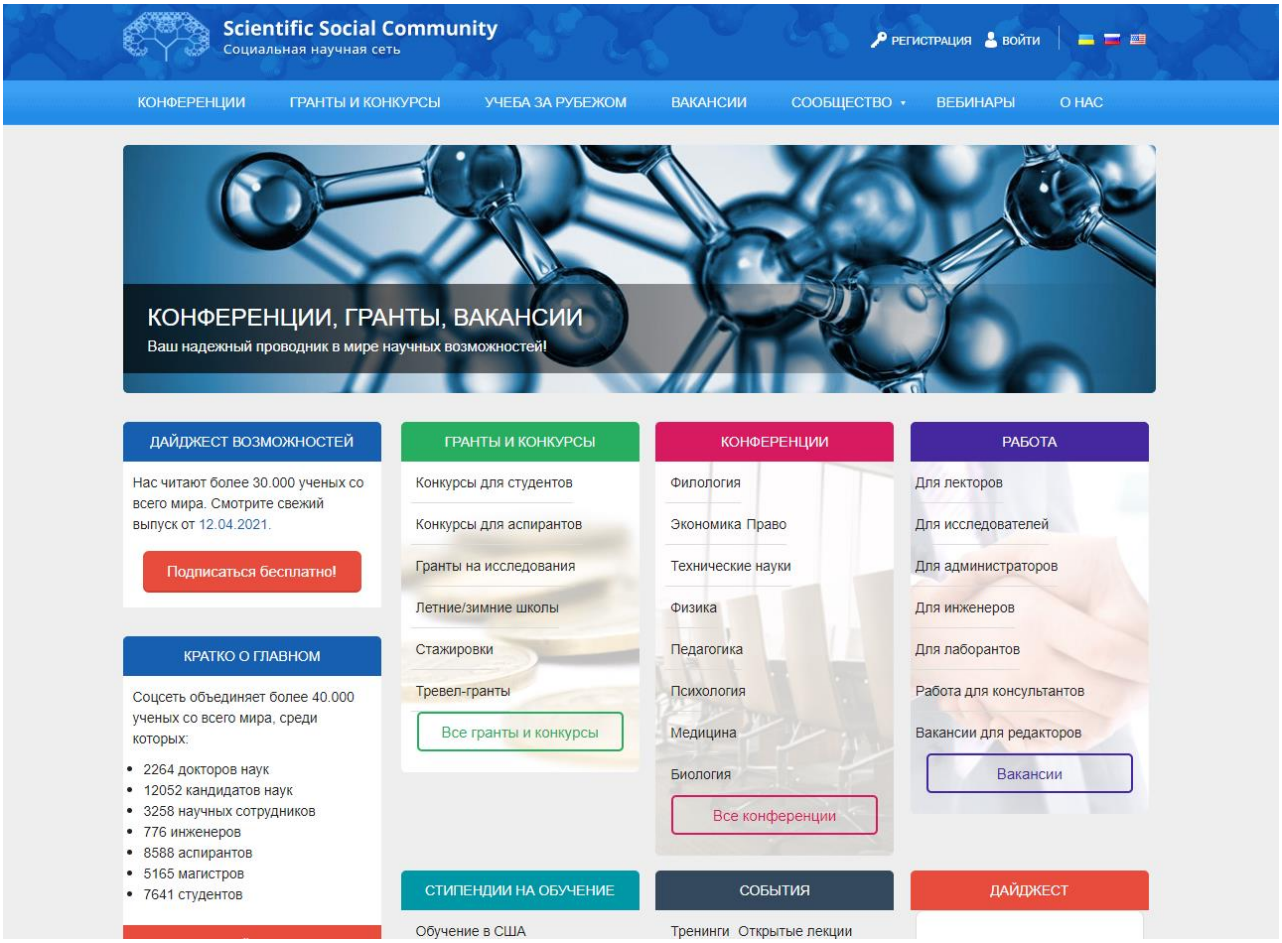

Рис 1.2 Стартова сторінка Scientific Social Community

Таким чином, Scientific Social Community надає актуальні інструменти для всіх учасників наукового процесу - і для студентів і випускників, і для молодих вчених, і для заслужених метрів науки. Завдяки цьому дану спільноту можна назвати перспективним і багатообіцяючим сервісом, який актуальний для якнайшвидшого розвитку науки в Україні.

Всі сервіси Scientific Social Community повністю безкоштовні, і для доступу до них вам досить просто пройти реєстрацію на сайті.

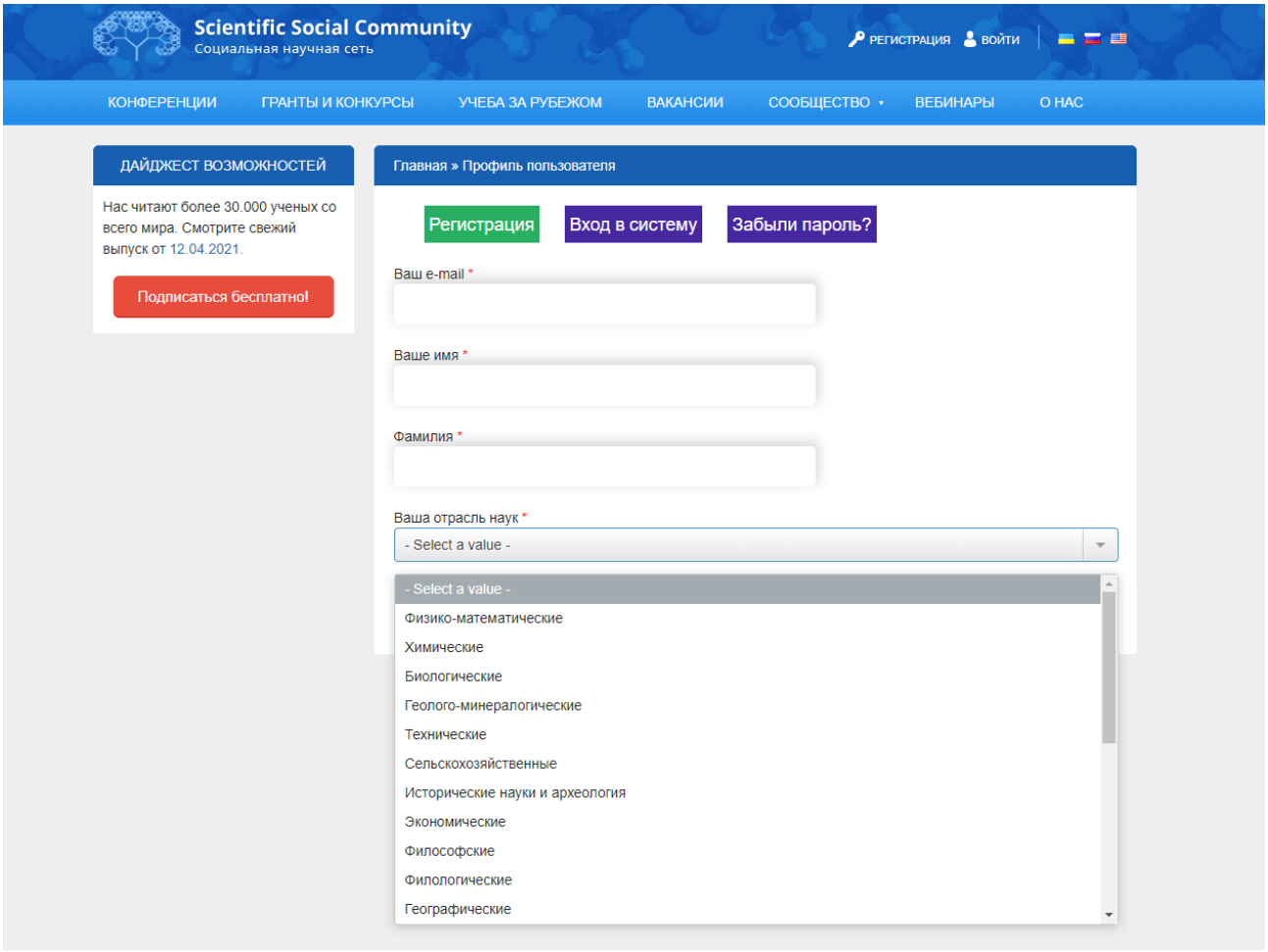

Рис 1.3 Форма реєстрації Scientific Social Community

Зареєстрованим користувачам доступна велика кількість грантів, пропозицій аспірантури і наукових вакансій, зібраних воєдино з численних розрізнених джерел, що також дуже актуально для українського наукового співтовариства.

Brainly

Brainly - багатонаціональний освітній стартап, що заснований в Польщі. Сайт призначений для студентів і викладачів. Він доступний тринадцятьма мовами світу і має більше 60 млн. унікальних користувачів щомісяця з більш ніж 35 країн світу. Це медіа-платформа для студентів, що стимулює спільне навчання. Вона використовує особливості соціальних мереж, щоб об'єднати користувачів, які хочуть ділитися своїми знаннями в рамках інтернет-спільноти.

Метою цього сайту є надихнути студентів вчитися в обстановці партнерства і розширити свої знання, орієнтуючись в першу чергу на виконання домашніх завдань.

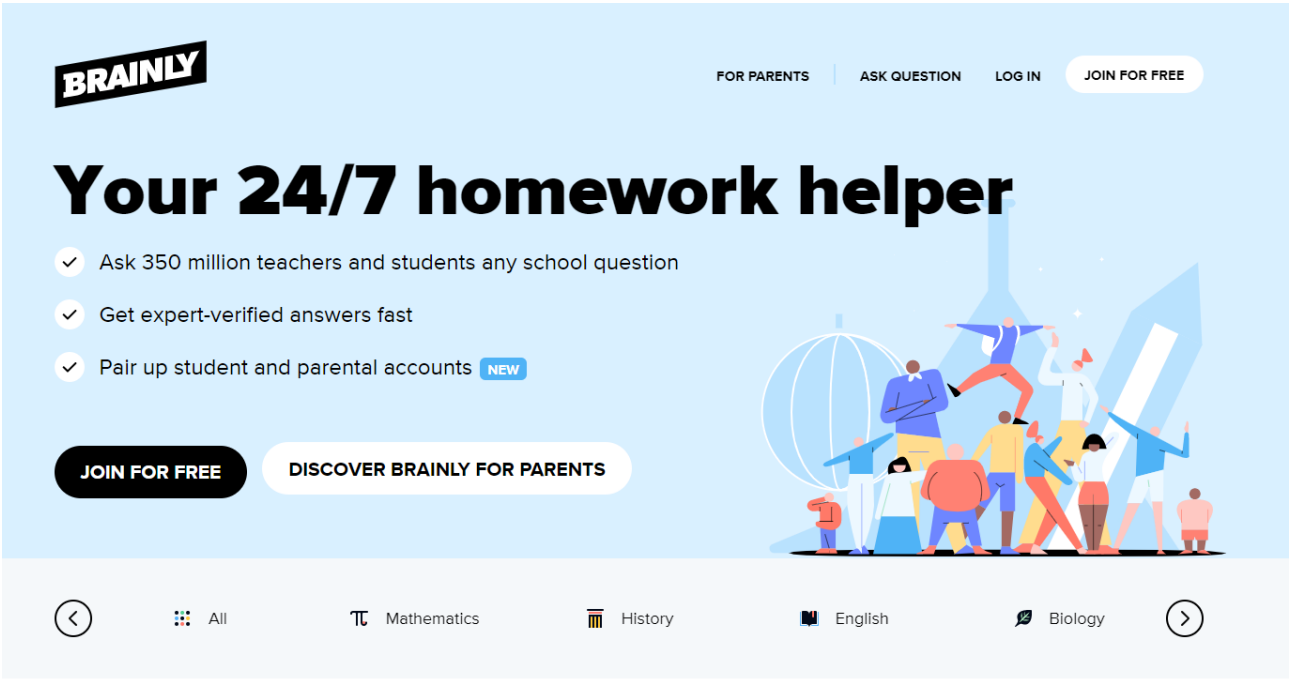

Рис 1.4 Стартова сторінка Brainly

Сайт присвячений студентам, яким потрібна допомога у виконанні домашніх завдань. Після реєстрації на сайті кожен може поставити питання або допомогти іншим. Користувачі можуть залишати свої коментарі на кожне питання і відповідь, також можуть вільно відповідати на поставлені запитання користувачів. Всі питання розподілені за темами, в залежності від дисципліни.

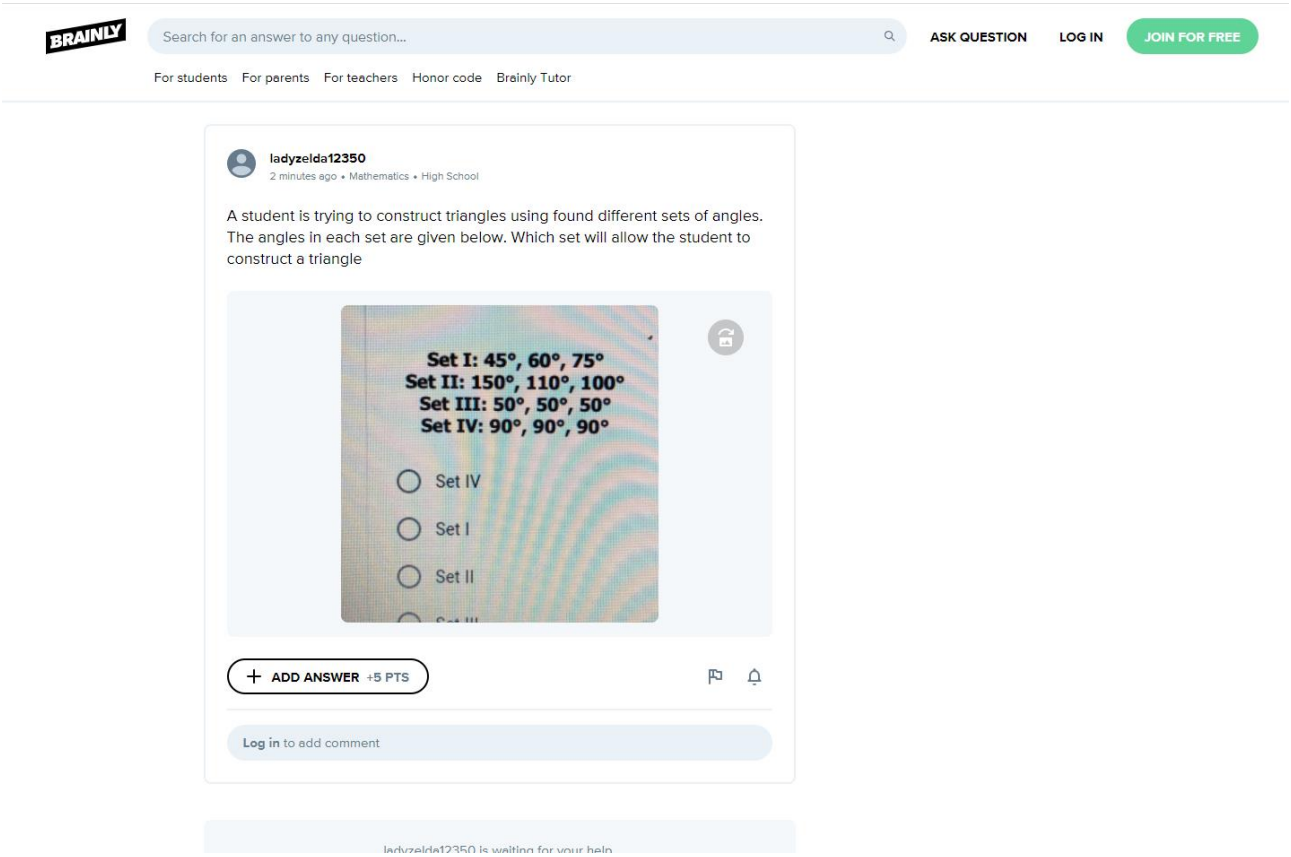

Рис 1.5 Приклад питання на сайті Brainly

Розглядувана соціальна мережа також містить елементи геймифікації, а саме:

Кожному користувачеві при реєстрації надаються бали з фіксованою сумою, які використовуються для запитання. Користувачі можуть набирати бали, відповідаючи на питання, розміщені іншими.

Brainly присуджує "ранги" користувачам, які надають часті якісні відповіді. Деякі звання автоматично винагороджуються за зароблену задану кількість балів або задану кількість найякісніших відповідей на запитання. Користувачі можуть також претендувати на "спеціальні звання", які можуть бути присвоєні за успіхи в конкретних предметах та інших критеріях. Крім того, веб-сайт присвоює рейтинг викладачів професійним учителям

На веб-сайті присутні користувачі, які відповіли на найбільшу кількість запитань або отримали найбільшу кількість балів за набір таблиць лідерів.

Таблиці лідерів класифікуються як щоденні, щотижневі, щомісячні та щоквартальні

Facebook

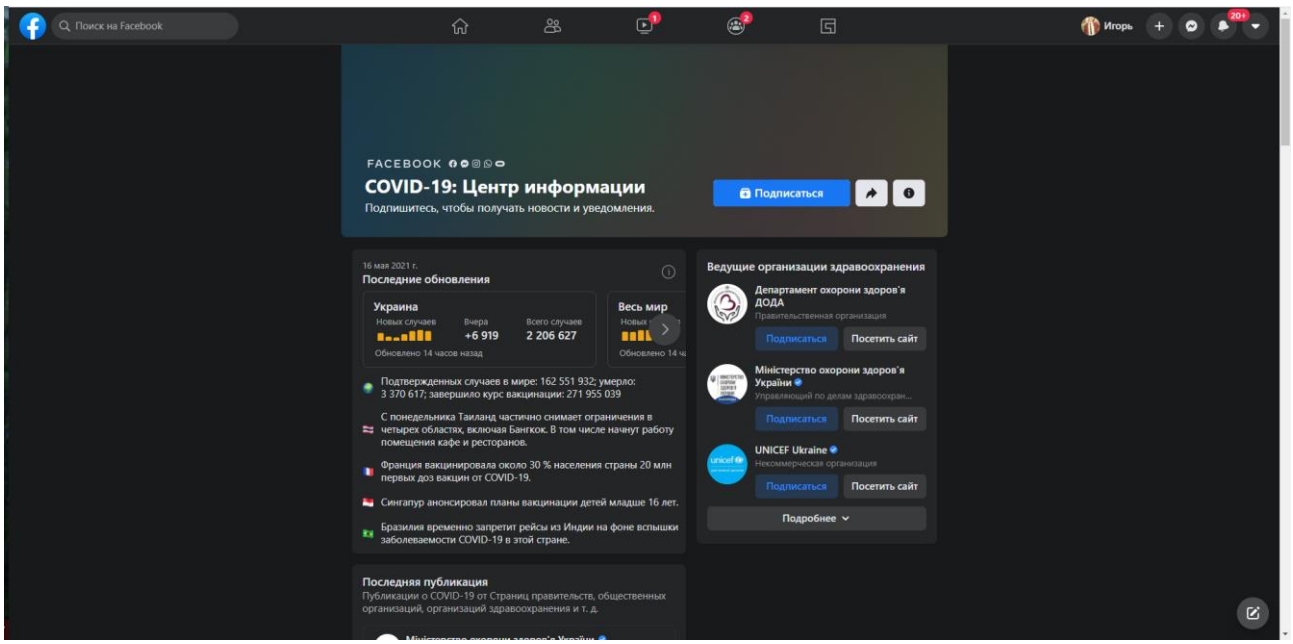

Рис. 1.6 Приклад сторінки соціальної мережі Facebook

Facebook - найбільша у світі соціальна мережа, що почала працювати 4 лютого 2004 року як мережа для студентів деяких американських університетів. Засновником та головою сервісу є Марк Цукерберг. За даними рейтингу Alexa Rank, сайт facebook.com посідає 7 місце за відвідуваністю у світі.

Користувачі Facebook мають можливість створювати профілі з фотографіями, списками інтересів, контактними даними та іншою особистою інформацією. Вони можуть спілкуватися із друзями та іншими користувачами за допомогою приватних або загальнодоступних повідомлень і чату. Також користувачі можуть створювати і приєднуватися до груп за інтересами та «сторінок уподобань» (до 19 квітня 2010 року вони називалися «фанат-сторінками»). Деякі з цих сторінок підтримують організації як засоби реклами. Згодом Facebook додав функції на свій вебсайт. 6 вересня 2006 року було випущено Стрічку новин, яка з'являється на сторінці кожного користувача і виділяє інформацію, в тому числі зміни профілю, найближчі події та дні народження друзів користувача[7].

Серед найпопулярніших Facebook-додатків:

– хостинг фотографій, де користувачі можуть завантажувати альбоми та фотографії. Facebook, на відміну від інших фотохостингів, таких як Photobucket і Flickr, дозволяє користувачам завантажувати необмежену кількість фотографій. У перші роки ліміт становив 60 фотографій для одного альбому, а з травня 2009 року він зріс до 200 фотографій.

– можливість проводити онлайн-трансляції. У червні 2017 року Facebook відкрив доступ усім користувачам до сервісу відеотрансляцій онлайн Facebook Live. Друзям надходить сповіщення про початок онлайн-трансляцій, є можливість ставити лайки, коментувати, а також поширювати відео онлайн.

– Facebook messenger - система обміну миттєвими повідомленнями, створена Facebook. Інтегрована з додатком на основному сайті Facebook і побудована на базі відкритого протоколу MQTT. Крім обміну повідомленнями, також використовується для обміну фотографіями, відео, аудіозаписами та для групових чатів. Додаток можна використовувати для спілкування зі своїми друзями у Facebook та з телефонними контактами.

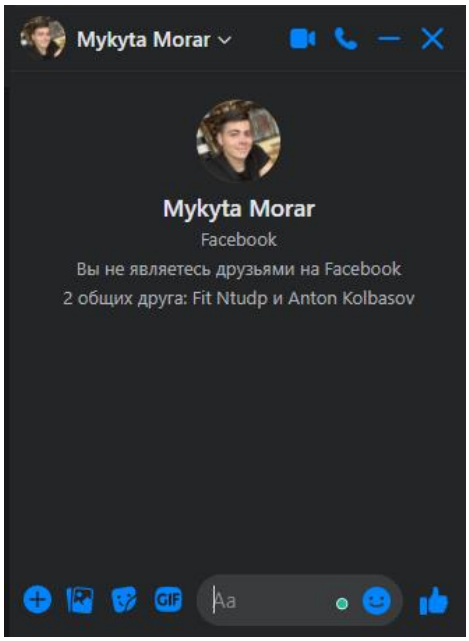

Рис. 1.7 Чат у Facebook messenger

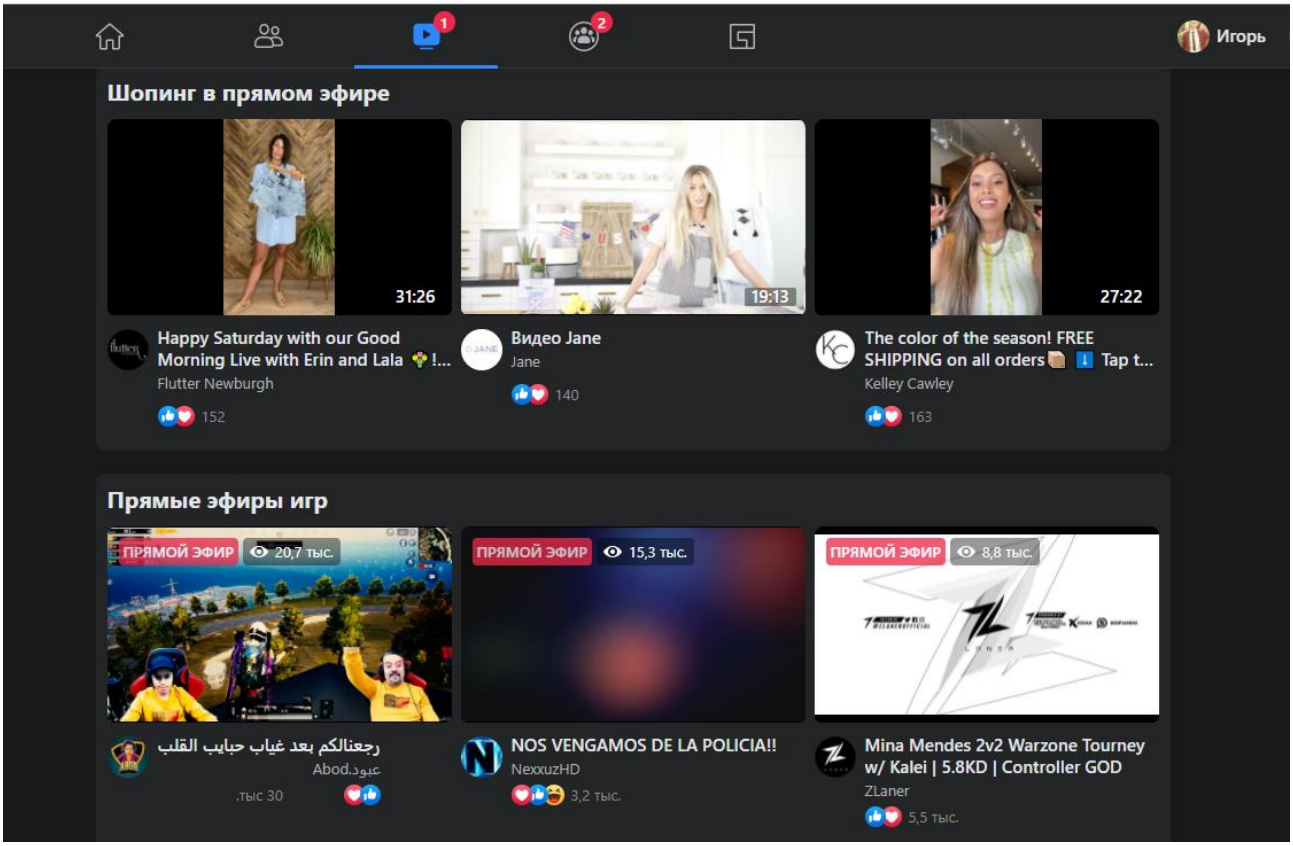

Рис. 1.8 Сторінка Facebook Live

1.2 Призначення розробки та галузь її застосування

Розробка універсальної серверної частини для соціальної мережі для студентів та викладачів.

Розроблений продукт має використовуватися студентами/викладачами для забезпечення комфортної та якісної взаємодії між собою, мінімізуючи затрачений час та зусилля.

Призначення розробки полягає наданні потенціальним користувачам наступних переваг автоматизованої системи:

– безперервна робота розроблюваної мережі

– дає можливість зареєструватися та використовувати розроблюваний мережу незалежно від місця навчання;

– дозволяє створювати студентські групи, дивитись розклад занять;

– дає можливість відстежувати заняття: присутність студентів, теми заняття, дата та час проведення;

– дозволяє здійснювати комунікацію за допомогою постів та коментарів;

– дає можливість будувати власне соціальне оточення через підписки на інших користувачів.

Таблиця 1.1

Переваги та недоліки програм

	Scientific Social Community	Brainly	Facebook	Розроблювана система
Можливість підписуватись на інших користувачів	Ні	Ні	Так	Так
Можливість створювати/редагувати та переглядати розклад	Ні	Ні	Так	Так
Можливість створювати групи	Ні	Так	Так	Так
Можливість переглядати історію проведених занять	Ні	Ні	Ні	Так
Можливість створювати публікації, коментувати їх та реагувати на них	Так	Так	Так	Так

З таблиці можливостей кожної з програм, можна зробити висновки, які можливості повинна надавати розроблювальна система, а яких недоліків вона повинна позбутися.

1.2 Підстави для розробки

Відповідно до освітньої програми, згідно навчального плану та графіків навчального процесу, в кінці навчання студент виконує кваліфікаційну роботу.

Тема роботи узгоджується з керівником проекту, випускаючою кафедрою, та затверджується наказом ректора.

Отже, підставами для розробки (виконання кваліфікаційної роботи) є:

– освітня програма 121 «Інженерія програмного забезпечення»;

– навчальний план та графік навчального процесу;

– наказ ректора Національного технічного університету «Дніпровська політехніка» № 317-с від 07.06 2021 р;

завдання на кваліфікаційну роботу на тему «Розробка універсальної серверної частини для соціальної мережі для студентів та викладачів».».

1.4. Постановка завдання

Завданням є розробка універсальної серверної частини для соціальної мережі для студентів та викладачів.

Реалізація поставленої мети передбачає вирішення наступних завдань:

Поставлена мета може бути досягнена при виконані наступних вимог:

– вивчення предметної області розв’язуваного завдання;

– проведення порівняльної характеристики можливостей аналогічних застосунків;

– проектування та розробка відповідної бази даних програми

– вибір релевантної архітектури програми;

– написання програмного коду застосунку;

Кінцева програма має представляти собою універсальний API, призначений для .

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Кінцевий продукт має дотримуватися наступних функціональними вимог:

– дані повинні вводитися користувачем у форму та передаватися на сервер у форматі JSON;

– при виникненні помилок, веб-сервіс повинен обробляти помилки та виводити правильні повідомлення з помилкою;

– користувач має можливість редагувати власні дані;

– має бути реалізована система постів та коментарів;

– сервер повинен обробляти інформацію та віддавати «відповідь» на кожен HTTP запит;

– повинна бути реалізована можливість підписуватись на інших користувачів;

– має бути можливість переглядати та встановлювати розклад занять;

– повинна бути можливість переглядати історію занять: теми, присутніх, виставлені оцінки;

– для викладачів має бути окремий функціонал, що дозволяє формувати студентські групи, додавати/видаляти звідти студентів;

– також має бути реалізований додатковий фунціонал для адміністраторів сайту, що будуть створювати сторінки вузів та додавати викладачів в дану систему;

– повинна бути реалізована можливість створення різних аккаунтів користувачів та повинна бути імплементована авторизація та аутентифікація;

1.5.2.Вимоги до інформаційної безпеки

Для забезпечення надійного функціонування системи необхідно реалізувати наступні вимоги:

– валідація введених даних: перевірка на відповідність типів, на введення обов’язкових полів даних;

– обробка виняткових ситуацій;

– виведення повідомлень у разі виникнення помилок;

– захист від несанкціонованого доступу до функціоналу за допомогою JWT (JSON WEB Token).

1.5.3. Вимоги до складу та параметрів технічних засобів

Для коректного функціонування кінцевого продукту, а саме автоматизованої системи управління та підтримки процесів онлайн продажів товарів електроніки необхідними технічними умовами є:

– операційна система Windows 10, Linux, MacOS, Android або IOS;

– обсяг оперативної пам’яті (ОЗУ) не менш 2 ГБ;

– наявність будь-якого сучасного браузера, наприклад, Google Chrome або Opera;

– підтримка високошвидкісного Інтернету.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

2.1. Функціональне призначення програми.

Основне призначення додатку складається з покращення взаємодії студентів та викладачів, шляхом розробки спеціальної соціальної платформи.

2.2. Опис застосованих математичних методів.

У розробленій системі не використовуються математичні методи.

2.3. Опис використаної архітектури та шаблонів проектування.

Шаблони проектування являють собою певний спосіб вирішення певної архітектурної проблеми. Тобто до них можна й потрібно відноситись, як до багаторічного досвіду проектування та побудови складних систем, що вийшов у набір основних прийомів та методів.

Патерни розрізняють за видами:

- породжуючі патерни;
- структурні патерни;
- патерни поведінки;

Для даного додатку було вибрано класична архітектура програмування REST:

REST (Representational State Transfer) — це стиль архітектури, що використовує HTTP, як протокол передачі даних. Системи, що використовують такий підхід, називаються RESTful – системами.

Перший, хто використав термін REST, був Рой Філдінг, котрий описав цей архітектурний стиль у своїй докторській дисертації[11]. Згідно цієї роботи, REST

– це загальний набір принципів побудови архітектури систем як у веб-просторі, так і у вбудованих систем, оскільки він не задає деталі реалізації.

Існує 6 загальних правил побудови REST-застосунку:

1. Клієнт-серверна модель

Віповідне розмежування обов’язків:

– Фронтенд частина додатку відповідає за візуальне оформлення та безпосередньо взаємодіє з клієнтом, відправляючи відповідні запити до бекенду

– Бекенд в свою чергу приймає ці запити, оброблює їх та відповідає фронтенду.

2. Кеширування

Збереження даних на стороні клієнта.

Інколи нема потреби кожен раз відсилати запит на бекенд. Для забезпечення швидкої роботи застосунку деякі результати запитів можна зберігати на стороні клієнта.

3. Відсутність стану

Відсутність збереженої інформації про клієнта на сервері. Усі запити супроводжуються шифрованими токенами доступу, які зберігають усю необхідну інформацію про клієнта.

4. Однорідний інтерфейс

Розроблюваний додаток має містити впорядковану структуру інтерфейсу. Енпоїнти мають бути описані в однорідному стилі, формат представлення має бути чітко визначений тощо

5. Шари абстракції

Поділ функціоналу на шари абстракції. Іншими словами – це декомпозиція та спрощення коду: кожен шар має працювати із відповідним йому функціоналом.

6. Запитування коду

Можливість завантаження деяких фрагментів коду у вигляди скриптів. Такий підхід хоч і «розвантажує» серверну частину, але це може бути небажаним с точки зору безпеки.

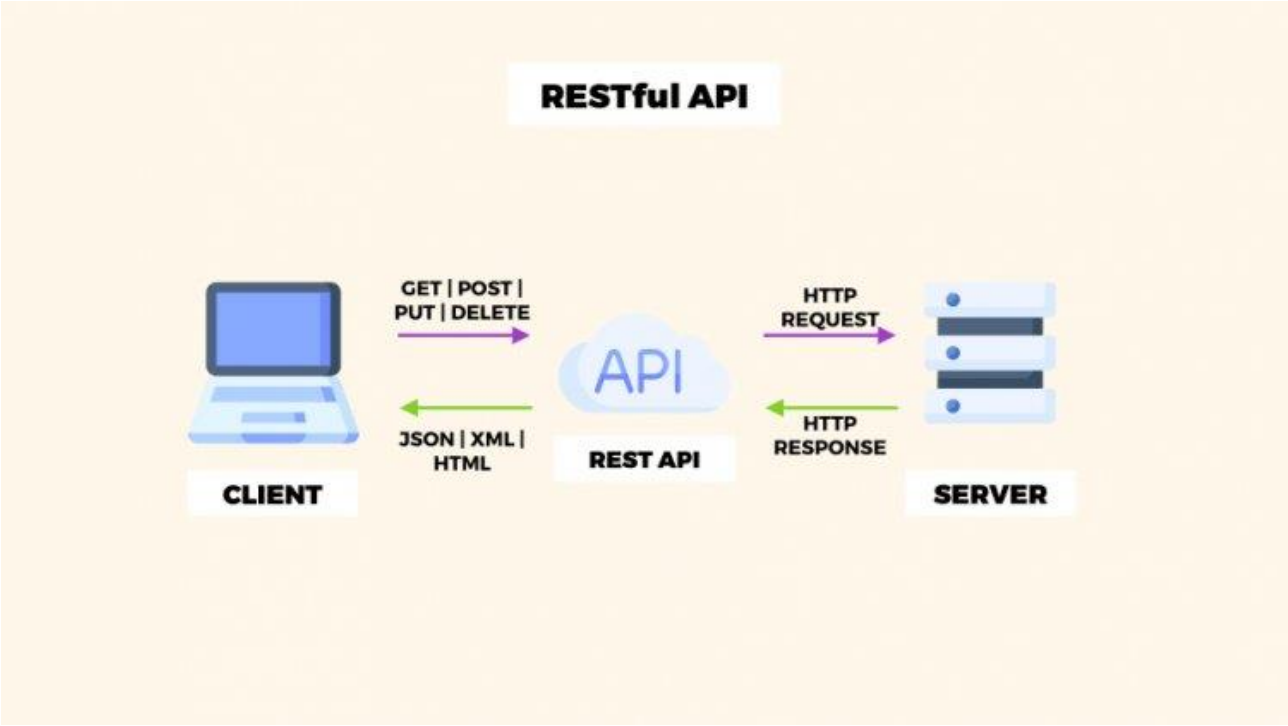

Рис. 2.1 Архітектура REST API

2.4. Опис використаних технологій та мов програмування.

Дана інформаційна система була розроблена з використанням таких технологій:

- Java
- Groovy
- Json Web Token
- Spring Boot
- Swagger
- PostgreSQL
- Liquibase
- Maven

Java

Java —об'єктно-орієнтована мова програмування, випущена В офіційній реалізації Java-програми компілюються у байт-код, який при виконанні

інтерпретується віртуальною машиною для конкретної платформи. Використовування даного язика не залежить від платформи.

Ключовою особливістю мови Java є те, що його код спочатку транслюється в спеціальний байт-код, сумісний із різними платформами. А потім цей байт-код виконується віртуальною машиною JVM (Java Virtual Machine). В цьому плані Java відрізняється від стандартних різних мов як Pyrhon або Ruby, код яких відразу ж виконується інтерпретатором. У той же час Java не є і чисто компільовані мовою, як C або C ++.

Подібна архітектура забезпечує кроссплатформенность і апаратну переносимість програм на Java, завдяки чому подібні програми без перекомпіляції можуть виконуватися на різних платформах - Windows, Linux, Mac OS і т.д. Для кожної з платформ може бути своя реалізація віртуальної машини JVM, але кожна з них може виконувати один і той же код.

Ще однією ключовою особливістю Java є те, що вона підтримує автоматичну збірку сміття. А це означає, що не треба звільняти вручну пам'ять від раніше використовувалися об'єктів, як в C ++, так як збирач сміття це зробить автоматично за вас[17].

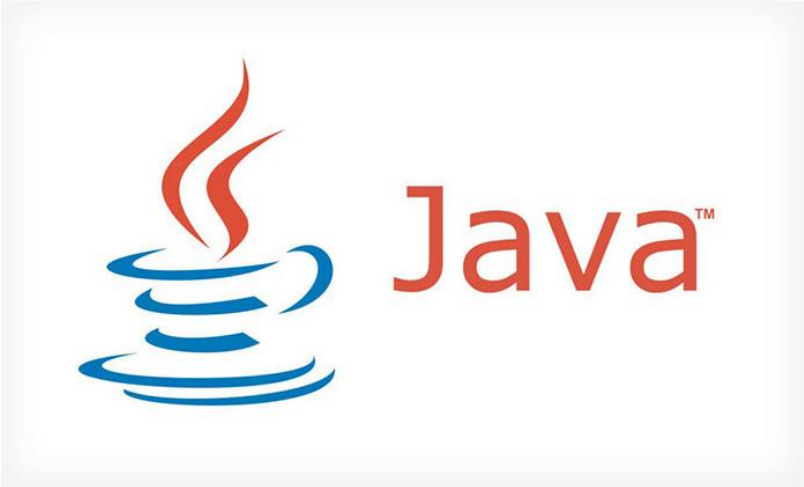

Рис. 2.2 Логотип Java

Плюси:

– ООП – мова програмування;

– Мультифункціональність;

– Надійність, завдяки строгій типізації;

– Крос-платформенність;

– Простий С-подібний синтаксис;

– Автоматична збірка сміття(видалення із пам’яті неактивних об’єктів, або об’єктів, на які немає посилань)

– Java має потужний інструментарій для розробки Android-застосунків;

Мінуси:

– Використовує багато ОЗУ

– Java – високорівнева мова програмування, не можна напряму керувати вказівниками.

– Швидкість порівняно менша, ніж у С та С++

Groovy

Groovy - об'єктно-орієнтована мова програмування, що розроблена для платформи Java як її альтернатива з можливостями Python, Ruby і Smalltalk.

Groovy використовує Java-подібний синтаксис з динамічної компіляцією в JVM байт-код і безпосередньо працює з іншим Java кодом і бібліотеками[16]. Мова може використовуватися в будь-якому Java проекті або як скриптова мова.

Можливості Groovy (що відрізняють його від Java):

– Статична і динамічна типізація

– Вбудований синтаксис для списків, асоціативних масивів, масивів і регулярних виразів

– Замикання(Clojure)

– Перевантаження операцій

Рис. 2.3 Логотип Groovy

Json Web Token

JSON Web Token — це стандарт токена доступу на основі JSON[8]. Використовується для верифікації тверджень.

Такий токен складається з трьох частин: заголовку, вмісту на підпису.

Заголовок(HEADER):

Це JSON елемент, що описує до якого типу належить даний токен і які методи шифрування використовувались.

Таблиця 2.1

Елементи JWT заголовку

Поле	Назва	Значення
typ	Type	JWT завжди мають медіатип ***application/jwt.***
cty	Content type	Це поле встановлюється коли JWT містить в собі інший JWT, в противному разі це поле пропускається.
alg	Algorithm	Описує використаний алгоритм шифрування. Зазвичай використовують HS256 або RS256. Можна також не використовувати жодного шифрування, вказавши none, але це не рекомендується.

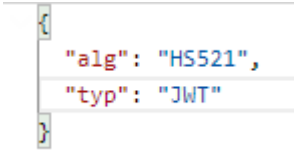

Рис. 2.4 Приклад заголовку JWT.

Вміст складається з елемента JSON який описує змінні, що задані користувачем. Вміст також має деякі необов’язкові зарезервовані значення:

Таблиця 2.2

Елементи JWT вмісту

Поле	Назва	Значення
Aud	Audience	Цільова аудиторія токена.

Exp	Expiration Time	Час закінчення терміну дії токена в форматі Unix (кількість секунд що пройшли від 1970-01-01T00:00:00Z).
Iat	Issued At	Коли токен був виданий (в форматі Unix)
Iss	Issuer	Той хто видав токен
Jti	JWT ID	Унікальна, чутлива до регістру послідовність символів, яка однозначно ідентифікує токен. Може застосовуватись для запобігання повторному використанню токена. Це можуть бути порядкові числа, GUID або Хеш.
Nbf	Not Before	Час в форматі Unix до настання якого токен не дійсний.
sub	Subject	Визначає якого суб'єкта стосуються твердження, тобто щодо кого або чого робляться твердження.

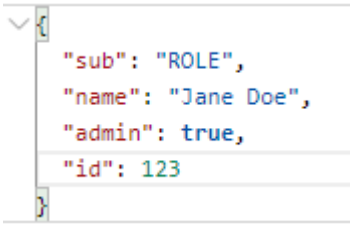

Рис. 2.5 Приклад вмісту JWT.

Підпис – остання частина токену, може бути довільної строкою, що задана користувачем, або ж генерується за допомогою заголовку та вмісту.

Заголовок, вміст і підпис кожен кодуються в Base64 і розділяються в токені крапкою. JWT може виглядати так:

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.**eyJpc3MiOiJzY290Y2guaW8iLCJleHAiOjEzMDA4MTkzdHJ1ZX0**.03f329983b86f7d9a9f5fef85305880101d5e302afafa20154d094b229f75773.

Тобто:

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9 – це закодований заголовок,

eyJpc3MiOiJzY290Y2guaW8iLCJleHAiOjEzMDA4MTkzdHJ1ZX0 – закодований вміст, та

03f329983b86f7d9a9f5fef85305880101d5e302afafa20154d094b229f75773 – відповідно закодований підпис.

Spring Boot

Spring Boot — це веб-фреймворк, що спрощує написання клієнт-серверних веб-застосунків за допомогою потужного вбудованого функціоналу[2].

Даний фреймворк може розглядатись як сукупність інших, взаємно незалежних фреймворків, котрі можуть як працювати окремо один від одного, так і разом, забезпечуючи максимальну ефективність роботи. Складові Spring діляться на наступні структурні елементи, серед яких найважливішими є:

– Inversion of Control-контейнер: конфігурування компонентів застосунку та керування життєвим циклом об’єктів.

– Фреймворк, що забезпечує безпеку застосунку: інструментарій конфігурації процесів автентифікації та авторизації, відомий як Spring Security.

– Фреймворк аспектно-орієнтованого програмування: компонент, відомий як AspectJ, за допомогою якого стає можливе так зване наскрізне програмування (можливість запроваджувати схожі блоки коду в різні місця програми під час її виконання).

– Фреймворк доступу даних: у Spring Boot таку роль виконує ORM Spring Data Jpa, за допомогою якої можна будувати інтуїтивно зрозумілі та прості запроси до серверу бази даних[4].

– Фреймворк для роботи з транзакціями: інструментарій налаштування транзацій.

– Фреймворк MVC: інструментарій для налаштування HTTP – запросів, що надає великий спектр можливостей для розробника.

– Тестування: Spring Boot надає широкі можливості для застосування різноманітних фрейморків тестування, таких як TestNG, Spock Framework, Junit тощо.

Порівняльна характеристика популярних веб-фреймворків:

Таблиця 2.3

Порівняльна таблиця можливостей різних веб – фреймворків.

Частина 1

Проект	Мова	Ajax	MVC - Framework	Інтер - націоналізація	ORM	Testing frameworks
Spring	Java	+	+	+	Hibernate, Jpa, iBatis.	Mockito, JUnit, TestNG
ASP.NET	C#	+	+	+	+	xUnit.net, MS Test, NUnit
Grails	Groovy	+	+	+	Hibernate, JPA, GORM	Spock Framework, JUnit
Django	Python	+	+	+	Django ORM	+

Таблиця 2.4

Порівняльна таблиця можливостей різних веб – фреймворків.

Частина 2

Проект	Мова	DB migration frameworks	Security frameworks	Teamplate frameworks	Caching frameworks	Validation frameworks
Spring	Java	-	Spring Security	JSP, Commons Tiles, Velocity, Thymeleaf	ehcache	Commons validator, Bean Validation
ASP.NET	C#	Entity Framework	ASP.NET Forms Authentication (Default), Pluggable	Razor (Default), ASPX, Pluggable	+	+

Grails	Groovy	multiple plugins: autobase, dbmigrate	Spring Security, Apache Shiro	+	+	+
Django	Python	Built into core	ACL-based	Django Template Language	Cache Framework	+

Swagger

Swagger – це фреймфорк специфікації REST-API, завдяки якому можна формалізувати існуючі ендпоінти та задокументувати створений API.

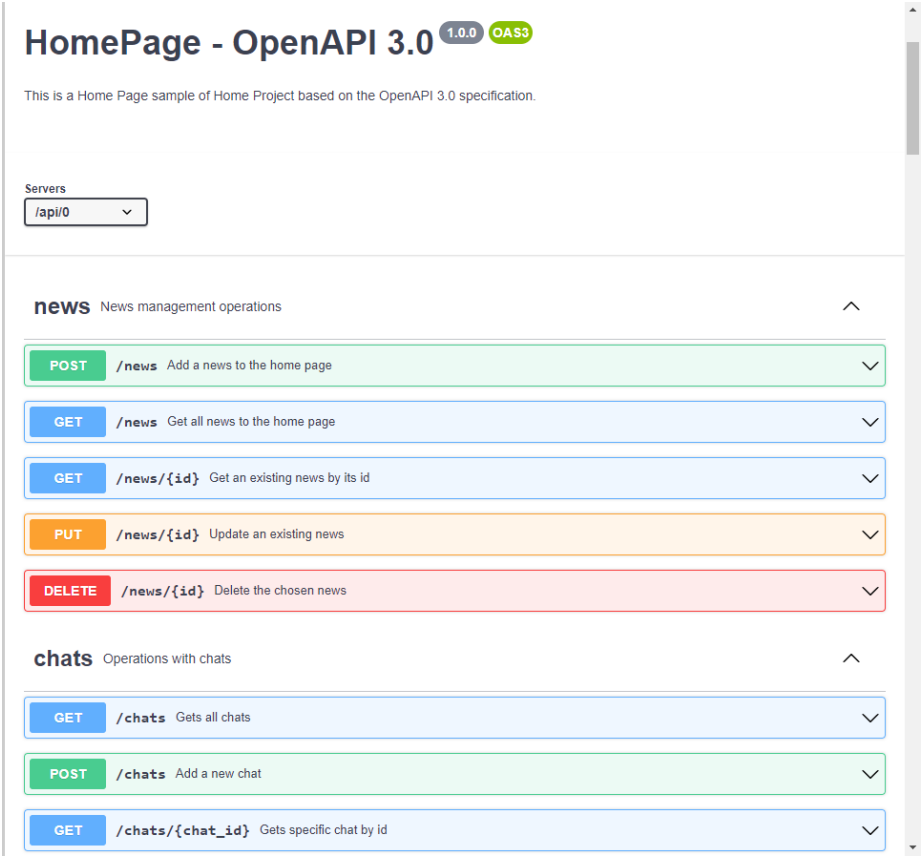

Рис. 2.6 Приклад OpenAPI специфікації.

Як можна бачити, ми можемо не тільки переглядати специфікацію, а й відправляти відповідні запити. Також є можливість сгенерувати сервер або клієнт, на основі існуючого конфігураційного файлу специфікації.

Swagger має два основних підходи написання документації:

– Code-first

Сутність цього підходу полягає у первинному написанні коду з подальшим його покриттям документації.

– API-first

Цей підхід має деякі переваги над попереднім, а саме:

– Уся документація буде суворо формалізована;

– Розробник з самого початку буде знати як правильно йому розробляти API

Але це не так просто, адже для написання специфікації треба знати синтаксис Swagger Specification та написати її однією з трьох мов (YAML, JSON або ж XML).

Liquibase

Liquibase - це бібліотека з відкритим кодом, незалежна від ядра БД, для відстеження, управління та застосування змін у схемі БД.

Усі зміни в базі даних зберігаються у текстових файлах (XML, YAML, JSON або SQL) та ідентифікуються за допомогою комбінації тегів "id" та "author", а також імені самого файлу. Список усіх застосованих змін зберігається у кожній базі даних, у таблиці DATABASECHANGELOG з якою проводиться консультація щодо всіх оновлень бази даних, щоб визначити, які нові зміни потрібно застосувати[18].

Liquibase автоматично генерує таблиці DATABASECHANGELOG та DATABASECHANGELOGLOCK під час першого застосування скриптів.

Із відомих аналогів бібліотек для міграції БД, що працюють з системами, розробленими на Java, існує FlywayDB.

Таблиця 2.5

Порівняння FlywayDB та Liquibase

	FlywayDB	Liquibase
Diff(порівняння двух баз даних)	-	+
Можливість ролбеку	+ (за додаткову плату)	+

Формати	SQL	SQL, XML, YAML, JSON
Можливість генерування SQL	-	+
Можливість застосовувати міграції в залежності від передумов	-	+

Maven

Maven - інструмент для автоматичної збірки проектів. З ним працюють в основному Java-розробники, хоча є плагіни для інтеграції з C / C ++, Ruby, Scala, PHP і іншими мовами.

Одна з головних особливостей фреймворка – декларативний опис проекту. Це означає, що розробнику не потрібно приділяти увагу кожному аспекту збірки - всі необхідні параметри налаштовані за замовчуванням. Зміни потрібно вносити лише в тому обсязі, в якому програміст хоче відхилитися від стандартних налаштувань.

Ще одна перевага проекту - гнучке управління залежностями. Maven вміє довантажувати в свій локальний репозиторій сторонні бібліотеки, вибирати необхідну версію пакету, обробляти транзитивні залежності.

Також даний фреймворк не залежить від ОС. При роботі з командного рядка параметри залежать від платформи, але Maven дозволяє не звертати уваги на цей аспект.

При необхідності систему збирання можна налаштувати під власні потреби, використовуючи готові плагіни. А якщо нічого підходящого не знайшлося - можна написати свої.

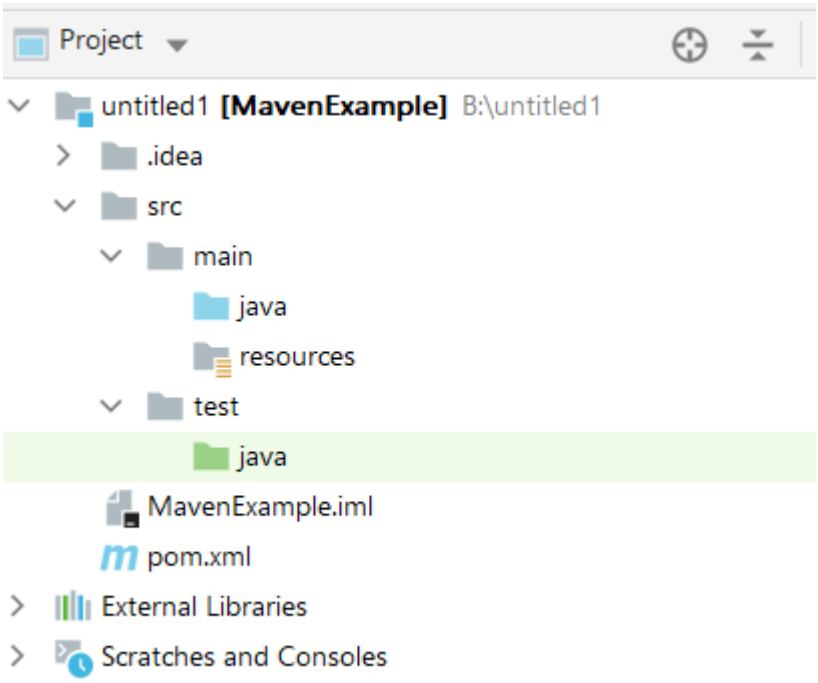

Рис 2.7 Структура проекту Maven

– src/main/java – директорія де лежать Java класи;
– src/main/resources – директорія для конфігураційних файлів;
– src/test/java – директорія де зберігаються тести.

2.5. Опис структури програми та алгоритмів її функціонування.

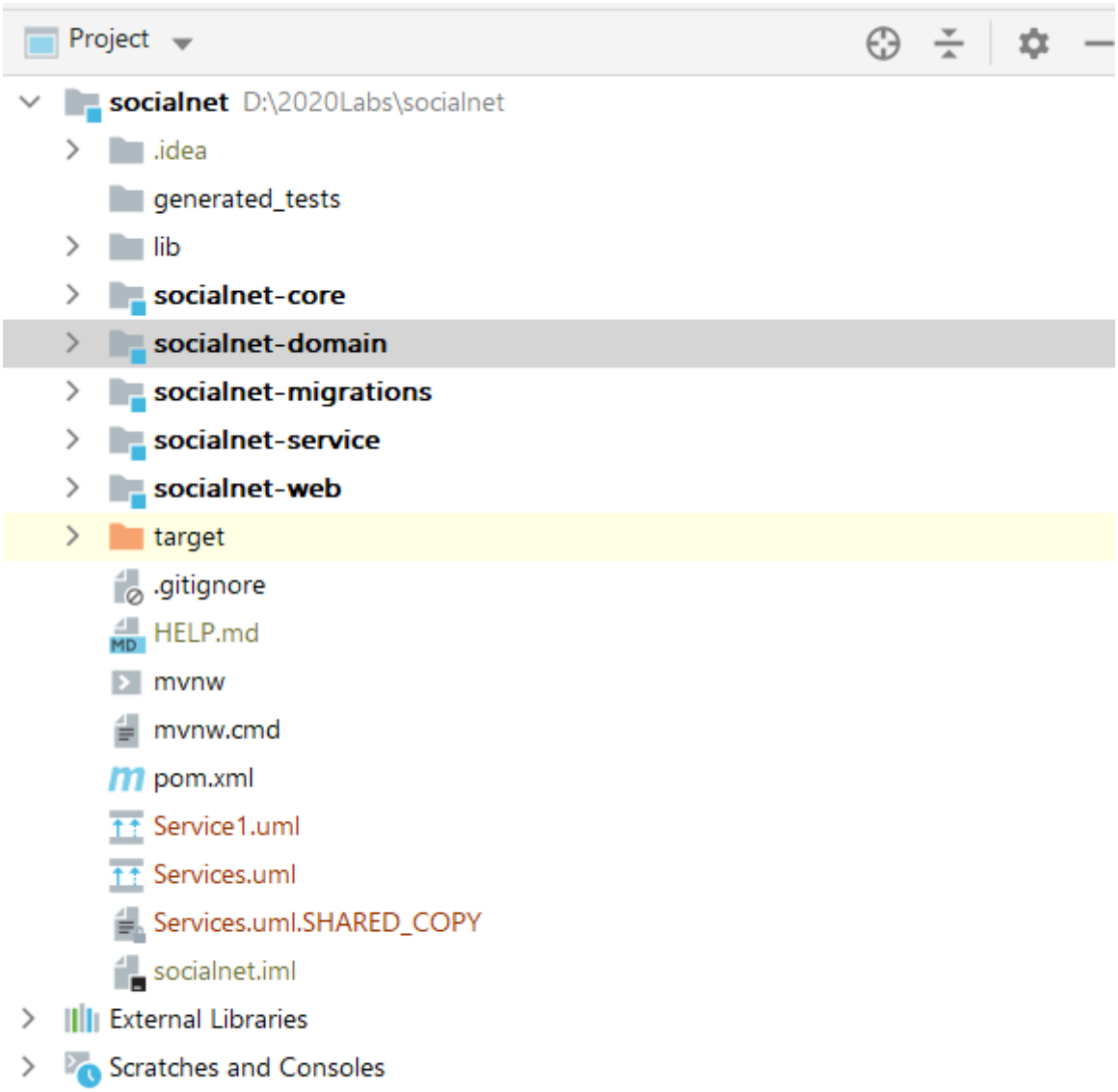

Рис. 2.8 Структура проекту

Дана програма побудована на основі мульти-модульної архітектури. Це означає додаток ділиться на декілька окремих модулів, що взаємодіють між собою, але мають різні сфери відповідальності.

Core:

Модуль Core – серце розроблюваного додатку, тут знаходяться класи конфігурацій, property-файли, та найважливіше – тут знаходиться точка входу застосунку – головний клас

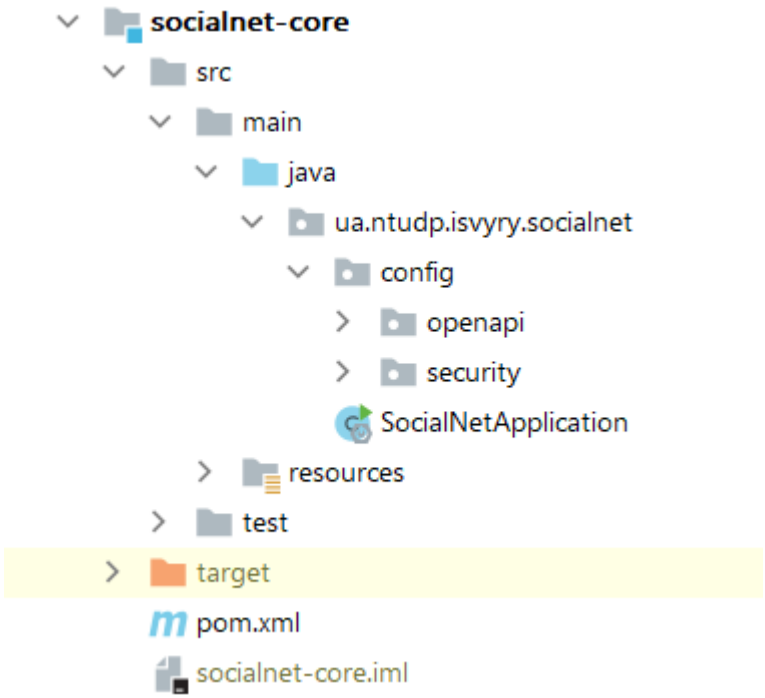

Рис. 2.9 Структура модулю Core

Domain:

Domain – модуль — це рівень взаємодії з базою даних. Тут знаходяться репозиторії та класи-моделі.

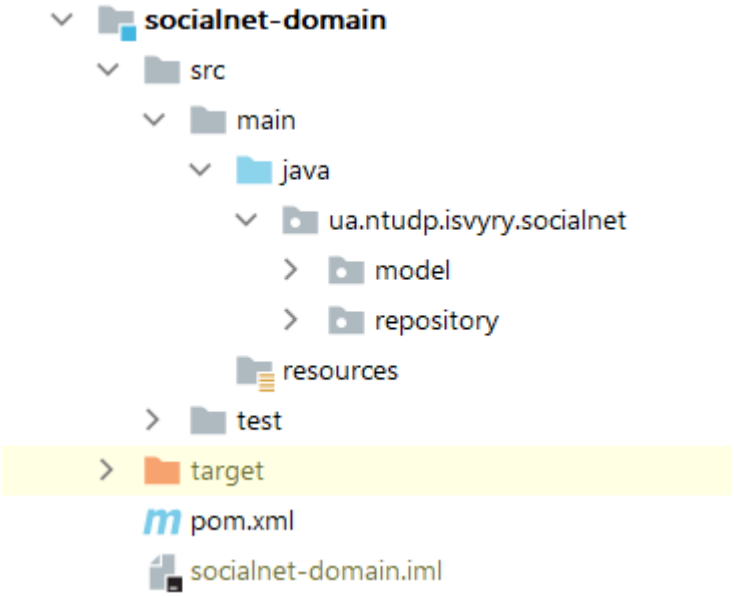

Рис. 2.10 Структура модулю Domain

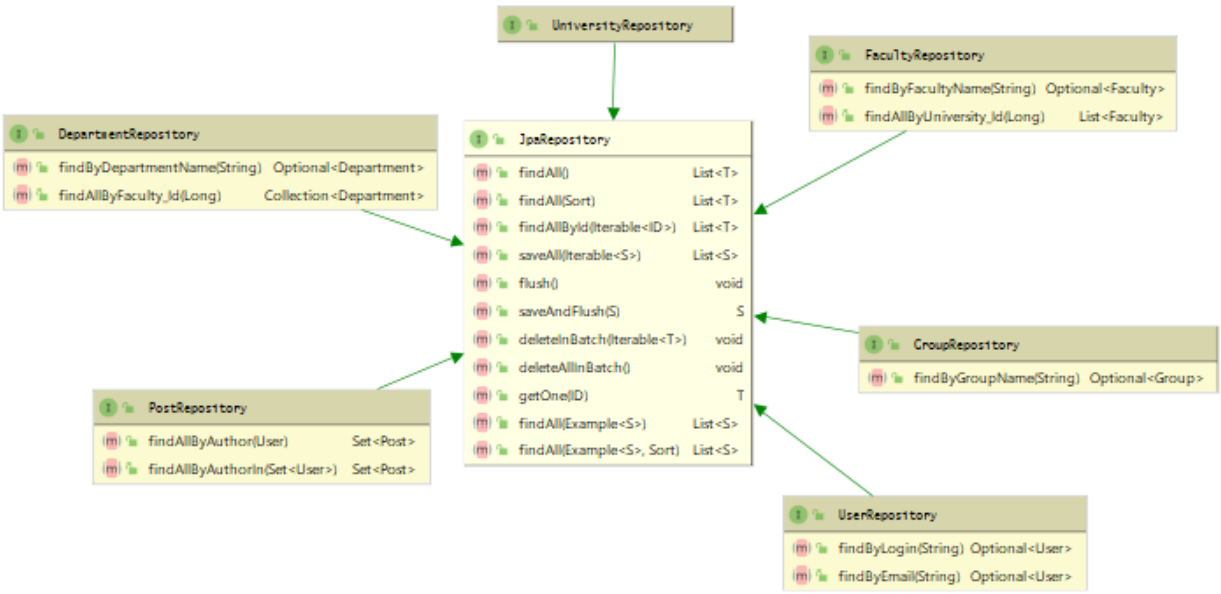

Рис 2.11 Найважливіші репозиторії. Частина 1

Як можна бачити на наведених діаграмах, усі репозиторії успадковують інтерфейс JpaRepository, який включає в себе найнеобхідніші методи для роботи з базою даних, такі як:

findAll(): знаходить усі записи в таблиці, еквівалентний запису «SELECT * FROM `*ENTITY*`»

deleteAll(): видаляє усі записи з таблиці, еквівалентний запису «DELETE FROM `*ENTITY*`»

deleteById(ID): видаляє з таблиці запис, що відповідає вказаному параметру, еквівалентний запису «DELETE FROM `*ENTITY*` WHERE ENTITY_ID = `*ID*`»

findById(ID): знаходить запис в таблиці, що відповідає вказаному параметру, еквівалентний запису «SELECT * FROM `*ENTITY*` WHERE ENTITY_ID = `*ID*`»

findAlllById(Collection<ID>): знаходить в таблиці всі записи, що відповідають критеріям пошуку, еквівалентний запису «SELECT * FROM `*ENTITY*` WHERE ENTITY_ID IN (Collection<ID>)»

saveAll(Collection<Entity>): зберігає або оновлює існуючі записи на основі вказаних параметрів.

Також, окрім стандартних реалізацій запитів, які нам надає Spring Data Jpa, існує можливість декларативним способом описати запити, які ми хочемо надіслати до бази даних.

Наприклад, ми хочемо знайти групу, згідно її назви. Для цього, у відповідному репозиторії, ми маємо задекларувати метод, у назві якого буде вказано поле, згідно якого буде побудований запит до БД, у нашому випадку це поле - groupName. Отже, наш метод буде виглядати наступним чином: findByGroupName(String groupName).

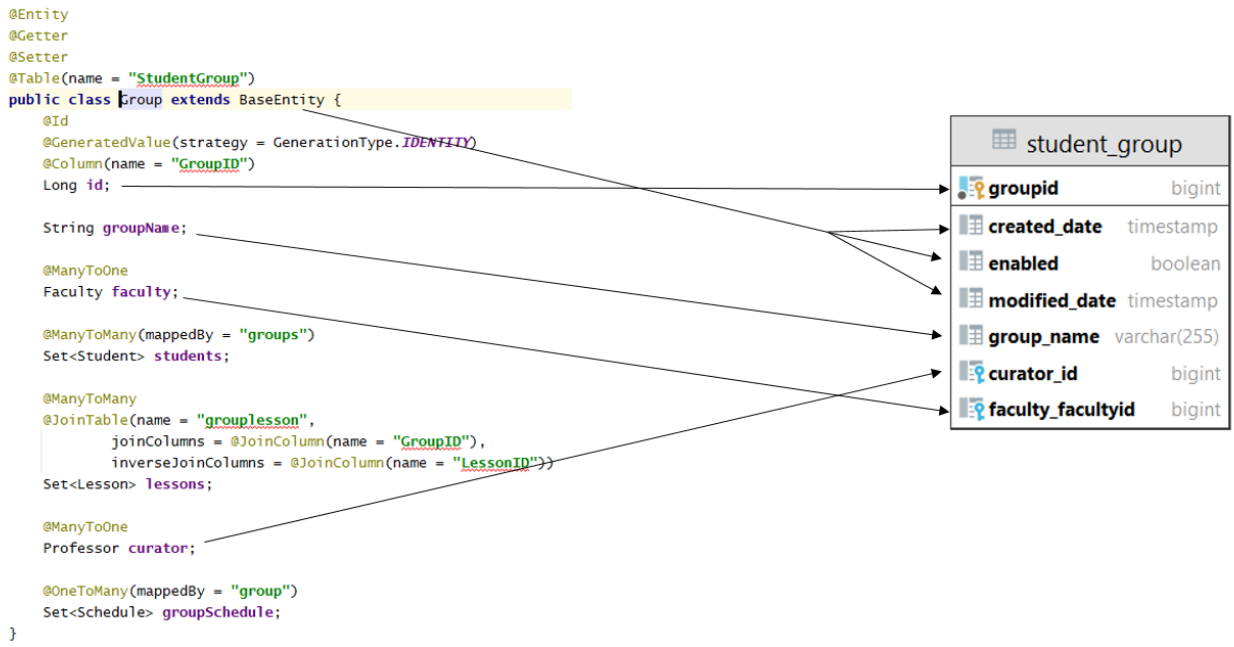

Рис. 2.12 Відповідність між властивостями класу та таблиці БД

Однак, інколи запити бувають дещо складнішими, котрі складаються із декількох нестандартних умов або з’єднань з іншими таблицями, такі запити зазвичай не можна прописати звичайним декларативним способом. У розроблюваному додатку таким прикладом є сутність «Розклад», адже даних тільки з відповідного репозиторію буде недостатньо для відображення повної інформації про розклад. Для цього небхідно створити власну реалізацію відповідного репозиторію, де ми за допомогою чистої мови SQL, власноруч прописуватимемо запити до БД, та перетворювати результат запиту до необхідної нам форми.

Наприклад, нам необхідно знайти розклад для групи з номером 122-17-1, для цього існує метод getScheduleByGroup(group: Group, scheduleType: Integer), де group – група, для якої буде знайдений розклад;

scheduleType – тип рокладу(1 – чисельник, 2 - знаменник).

```
Map<DayOfWeek, List<ScheduleSlimTransformer>> getScheduleByGroup(Group group, Integer scheduleType) {

    List<SocialNetScheduleTransformer> list = sessionFactory.currentSession.createSQLQuery("""select sc.day as day, d.discipline_name  as discipline, c.lesson_number as le
            left join student_group as sg
            on sg.groupid=sc.group_groupid
            left join discipline as d                                                    (1)
            on d.disciplineid = sc.discipline_disciplineid
                        join class as c on c.id= sc.a_class_id
            and sg.group_name = :groupName(2)
            and sc.type = :scheduleType(2)
            order by sc.day""")
            .setString("groupName", group.getGroupName()) (2)
            .setInteger("scheduleType", scheduleType) (2)
            .setResultTransformer(ScheduleTransformer.INSTANCE) (3)
            .list()(4)

    Map map = list.groupBy { SocialNetScheduleTransformer s -> s.day }(5)
    Map newMap = map.collectEntries { Map.Entry<DayOfWeek, List<SocialNetScheduleTransformer>> entry ->
        [ (entry.getKey()) : entry.getValue().collect { s ->
            new ScheduleSlimTransformer(s.disciplineName, s.lessonNumber, s.startTime, s.endTime) } ]   (6)
    }
    newMap
}
```

Рис 2.13 Запит для знаходження розкладу

Давайте розберемося що відбувається в даному методі:

1. Створення SQL-запиту до БД.
2. Встановлення параметрів у створений запит.
3. Вказівка на той клас, екземпляр якого має ввібрати результат запиту.
4. Виконання запиту.
5. Групування результуючого списку за днем тижня.
6. Створення відповідної карти елементів, ключом якої є день тижня, а значенням – список дисциплін, що встановлені у розкладі в цей день, час їх проведення та номер уроку.

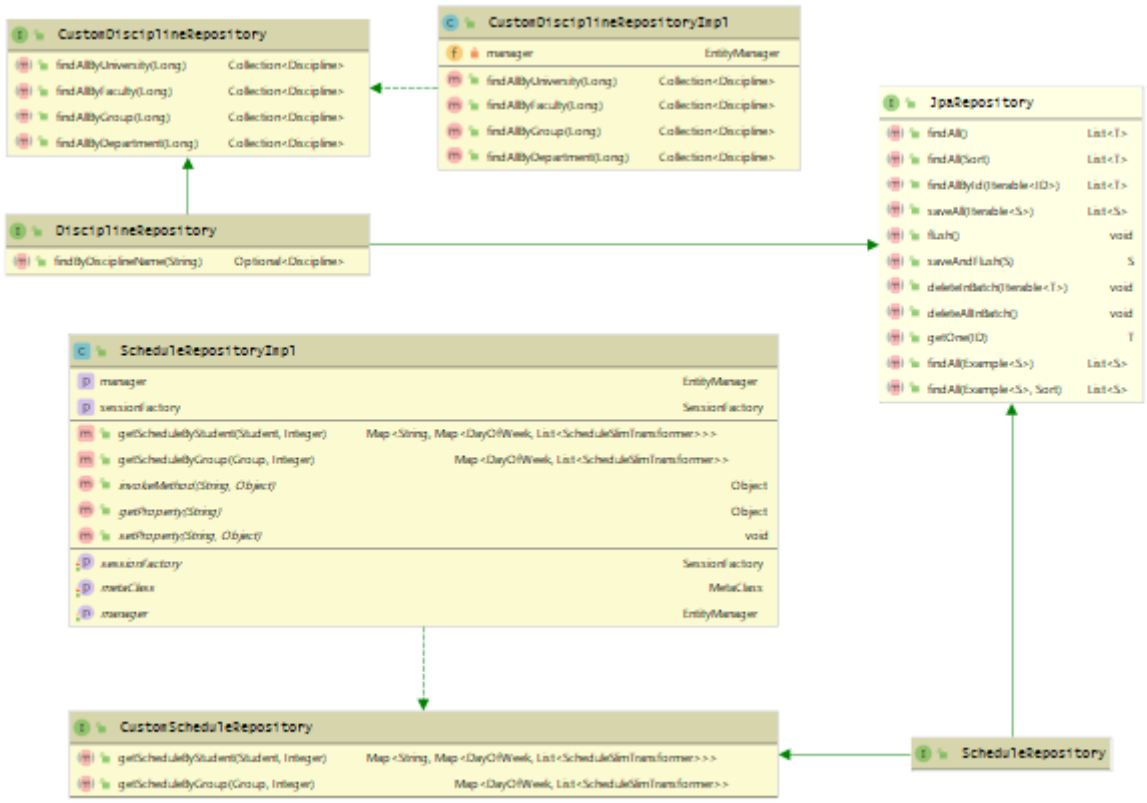

Рис 2.14 Найважливіші репозиторії. Частина 2

Migrations:

Даний модуль відповідає за маніпуляцію базою даних. Тут зберігаються файли, що звуться ченжлогами. В таких файлах можна створювати таблиці, наповнювати їх даними, також у цих файлах можна створювати різноманітні об’єкти БД, такі як: індекси, тригери тощо. Формати запису: .xml, .sql

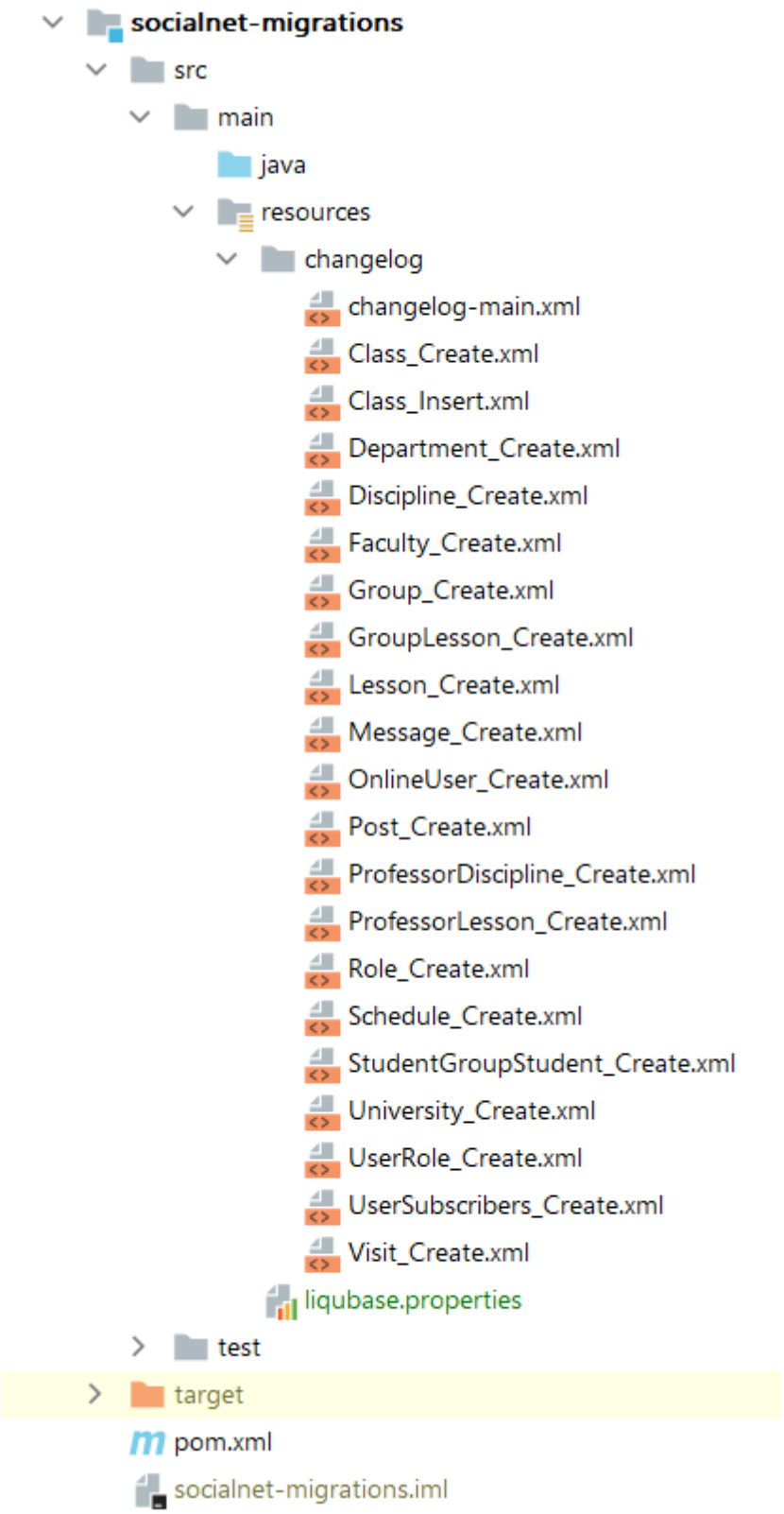

Рис. 2.15 Структура модулю Migrations

Service:

Модуль Service відповідає за бізнес-логіку застосунку. Також тут зосереджені конвертери, написані розробником виключення та так звані DTO об’єкти.

DTO (Data transfer object) – об’єкти передачі даних, не несуть у собі ніякої логіки, а використовуються у якості об’єктів, що взаємодіють із web – шаром.

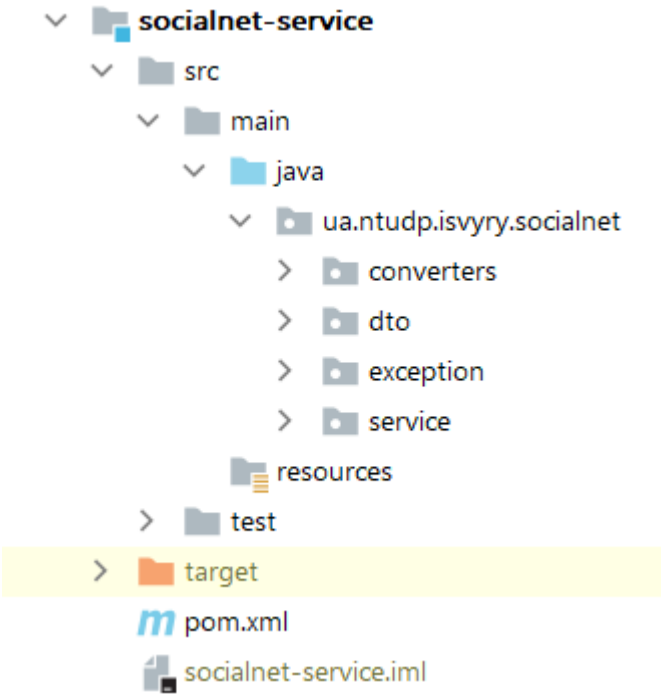

Рис. 2.16 Структура модулю Service

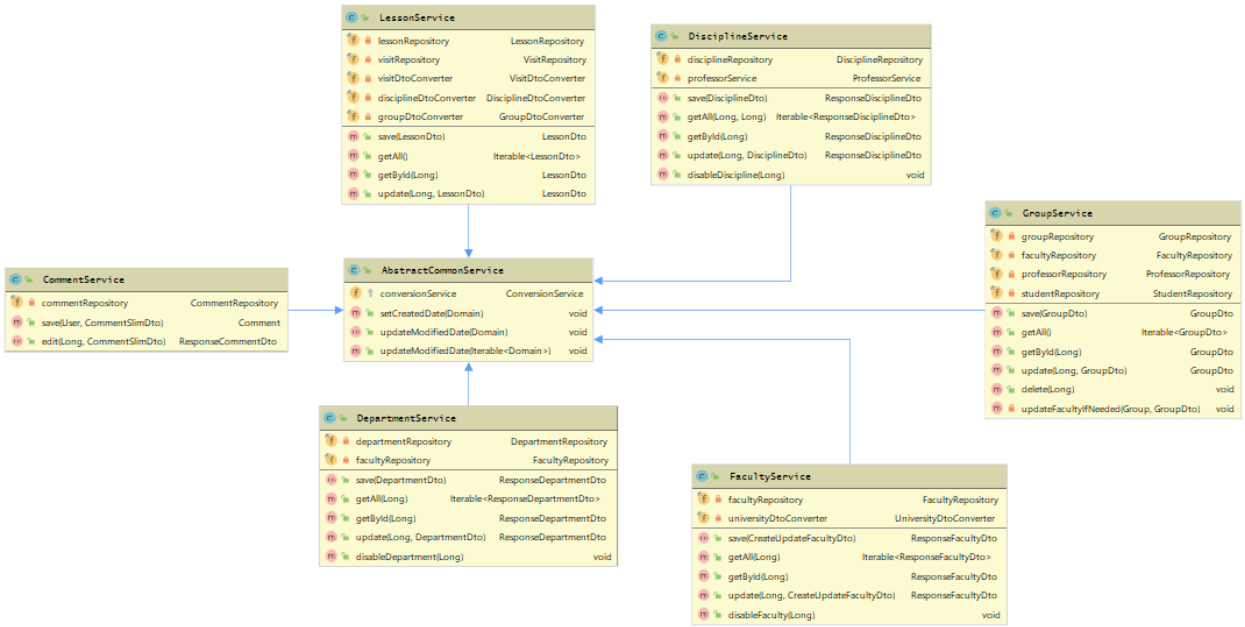

Рис. 2.17 Найважливіші сервіси. Частина 1

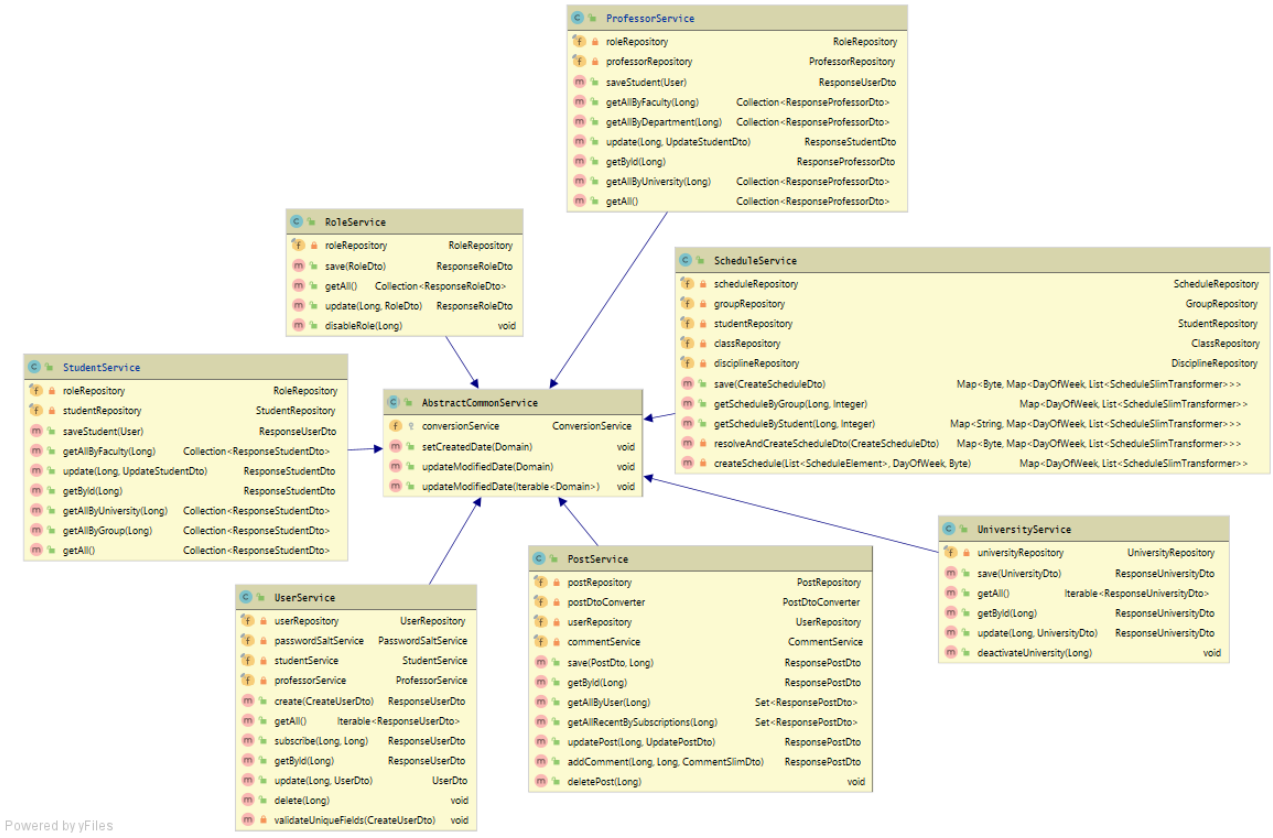

Рис. 2.18 Найважливіші сервіси. Частина 2

Таблиця 2.6

Функції сервісів

Сервіс	Призначення
UserService	Використовується для керування користувачами. А саме: створення, підписування на іншого користувача, знаходження користувачів за певним фільтром та деактивація користувача.
RoleService	Даний сервіс використовується для керування ролями. У ньому доступний функціонал додавання нової ролі, отримання всіх доступних ролей, зміна властивостей ролі.
PostService	Має контроль над функціоналом публікацій. Дозволяє створювати,

	редагувати, читати та видаляти публікації
ScheduleService	Використовується для маніпулювання розкладом. Даний сервіс дає змогу дивитися розклад відносно групи або окремого студенту, міняти та створювати розклад
GroupService	Дає змогу створювати, редагувати, знаходити та видаляти групи.
LessonService	Використовується для створення та редагування минувших уроків та їх відвідань.
UniversityService	Дають змогу створювати, редагувати, знаходити за певним фільтром та деактивувати відповідно університети, факультети та кафедри.
FacultyService	
DepartmentService	

Як можна бачити з UML-діаграм, кожен сервіс, що відповідає за певну бізнес-логіку, та безпосередньо взаємодіє з відповідним репозиторієм спадкує AbstractCommonService, який включає в себе ConversionService, який використовується в кожному сервісі, та має методи, що спільні для всіх сервісів-спадкоємців. Також кожен з цих сервісів має такі спільні методи, як save(), update(), getById(). getAll(), delete() – що забезпечують увесь необхідний CRUD-функціонал.

Але варто окреслити методи, що виділяються поміж загальних.

ScheduleService:

getScheduleByGroup() – використовується для отримання розкладу для певної групи.

getScheduleByStudent() – використовується для отримання розкладу для вказаного студенту.

getScheduleByGroup() приймає наступні параметри:

group: Group – група, для якої потрібно сформувати розклад

scheduleType: Integer – тип розкладу (1 – чисельник, 2 – знаменник)

getScheduleByGroup() віддає карту елементів. Де ключом є день тижня, а значення – список дисциплін.

getScheduleByStudent() приймає наступні параметри:

student: Student– студент, для якого потрібно сформувати розклад

scheduleType: Integer – тип розкладу (1 – чисельник, 2 – знаменник)

getScheduleByStudent() віддає карту елементів. Де ключом є назва групи, а значення – карта з розкладом, де в свою чергу ключом є день тижня , а значення список відповідних дисциплін.

PostService:

getAllRecentBySubscriptions() – знаходить існуючі пости тих користувачів, на яких підписан поточний.

addComment() – додає коментар до вказаної публікації.

getAllRecentBySubscriptions() приймає наступні параметри:

userId: Long – ідентифікатор користувача, згідно підписків якого будуть знайдені публікації.

getAllRecentBySubscriptions() віддає колекцію публікацій.

addComment() приймає наступні параметри:

id: Long - ідентифікатор публікації, для якої залишили коментар.

userId: Long - ідентифікатор користувача, що залишив коментар.

commentDto: CommentDto – тіло запиту, що несе інформацію про створений коментар

addComment() віддає оновлений пост.

UserService:

subscribe() – цей метод здійснює процес підписки на оновлення користувача.

subscribe() приймає наступні параметри:

currentUserId: Long - ідентифікатор поточного користувача.

userId: Long - ідентифікатор користувача, підписка на якого здійснюється.

subscribe() віддає оновленого користувача.

ProfessorService:

getAllByUniversity() – знаходить викладачів за вказаним університетом.

getAllByFaculty() - знаходить викладачів за вказаним факультетом.

getAllByDepartment() - знаходить викладачів за вказаною кафедрою.

getAllByUniversity() приймає наступні параметри:

universityId: Long – ідентифікатор університету, за яким здійснювати пошук.

getAllRecentByUniversity() віддає колекцію викладачів.

getAllByFaculty() приймає наступні параметри:

facultyId: Long – ідентифікатор факультету, за яким здійснювати пошук.

getAllByFaculty() віддає колекцію викладачів.

getAllByDepartment() приймає наступні параметри:

departmentId: Long – ідентифікатор кафедри, згідно якої здійснювати пошук.

getAllByDepartment() віддає колекцію викладачів.

Web:

Даний програмний шар в свою чергу містить контролери та різні конфігураційні класи, що працюють із запитами на сервер на відповідями.

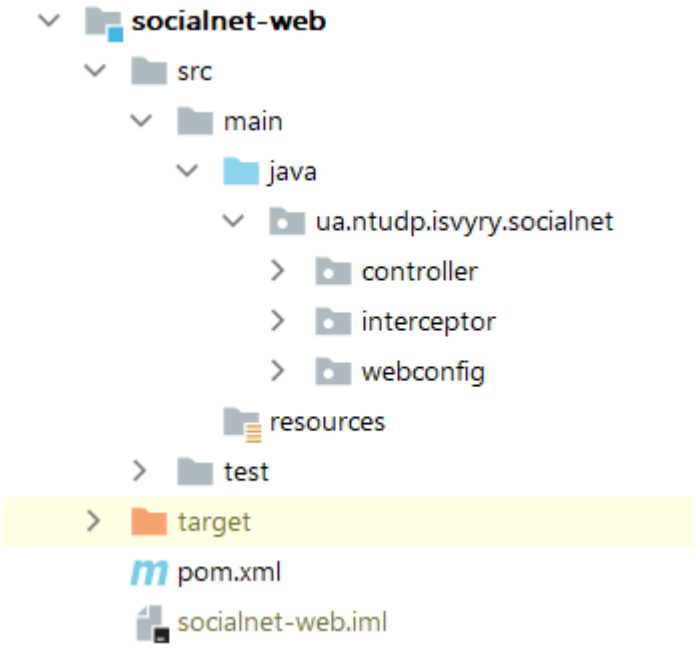

Рис. 2.19 Структура модулю Web

Таблиця 2.7

Найважливіші запити

Контроллер	Запит	Тип запиту	Призначення
Auth	/api/auth	POST	Використовується для авторизації користувача
	/api/logout	DELETE	Використовується для виходу користувача із застосунку
Post	/api/posts	POST	Створення публікації
	/api/posts/{id}	GET	Знаходження публікації
	/api/posts/users/{id}	GET	Знаходження публікацій відносно користувача
	/api/posts/{id}/comments	PUT	Редагування публікації(додавання коментаря)
	/api/posts/{id}	PUT	Редагування публікації
	/api/posts/users/{id}/subscription	GET	Знаходження публікацій відносно підписок поточного користувача
User	/api/users	POST	Створення користувача
	/api/users	GET	Знаходження всіх користувачів
	/api/users/{id}	PUT	Знаходження користувача відносно ідентифікатора
	/api/users/{id}	DELETE	Деактивація користувача
	/api/users/{id}/subscribe/{sId}	PUT	Додавання користувача до підписок
Role	/api/roles	POST	Створення ролі
	/api/roles	GET	Знаходження всіх існуючих ролей

	/api/roles/{id}	PUT	Редагування ролі
	/api/roles/{id}	DELETE	Деактивація ролі
Schedule	/api/schedules	POST	Створення розкладу
	/api/schedules	PUT	Редагування розкладу
	/api/schedules/groups/{id}	GET	Знаходження розкладу відносно групи
	/api/schedules/students/{id}	GET	Знаходження розкладу відносно студенту
Group	/api/groups	POST	Створення групи
	/api/groups	GET	Знаходження всіх груп
	/api/groups/{id}	GET	Знаходження групи за ідентифікатором
	/api/groups/{id}	PUT	Редагування групи
	/api/groups/{id}	DELETE	Деактивація групи
Faculty	/api/faculties	POST	Створення факультету
	/api/faculties	GET	Знаходження всіх факультетів
	/api/faculties/{id}	GET	Знаходження факультету за ідентифікатором
	/api/faculties/{id}	PUT	Редагування факультету
	/api/faculties/{id}	DELETE	Деактивація факультету
Lesson	/api/lessons	POST	Створення уроку
	/api/lessons	GET	Знаходження всіх уроків
	/api/lessons/{id}	GET	Знаходження уроку за ідентифікатором
	/api/lessons/{id}	PUT	Редагування уроку
Professor	/api/professors/{id}	PUT	Редагування користувача із роллю «викладач»
	/api/professors	GET	Знаходження користувачів із роллю «викладач»
	/api/professors/{id}	GET	Знаходження викладача за ідентифікатором

	/api/professors/universities/{id}	GET	Знаходження викладачів за вказаним університетом
	/api/professors/faculties/{id}	GET	Знаходження викладачів за вказаним факультетом
	/api/professors/groups/{id}	GET	Знаходження викладачів за вказаною групою

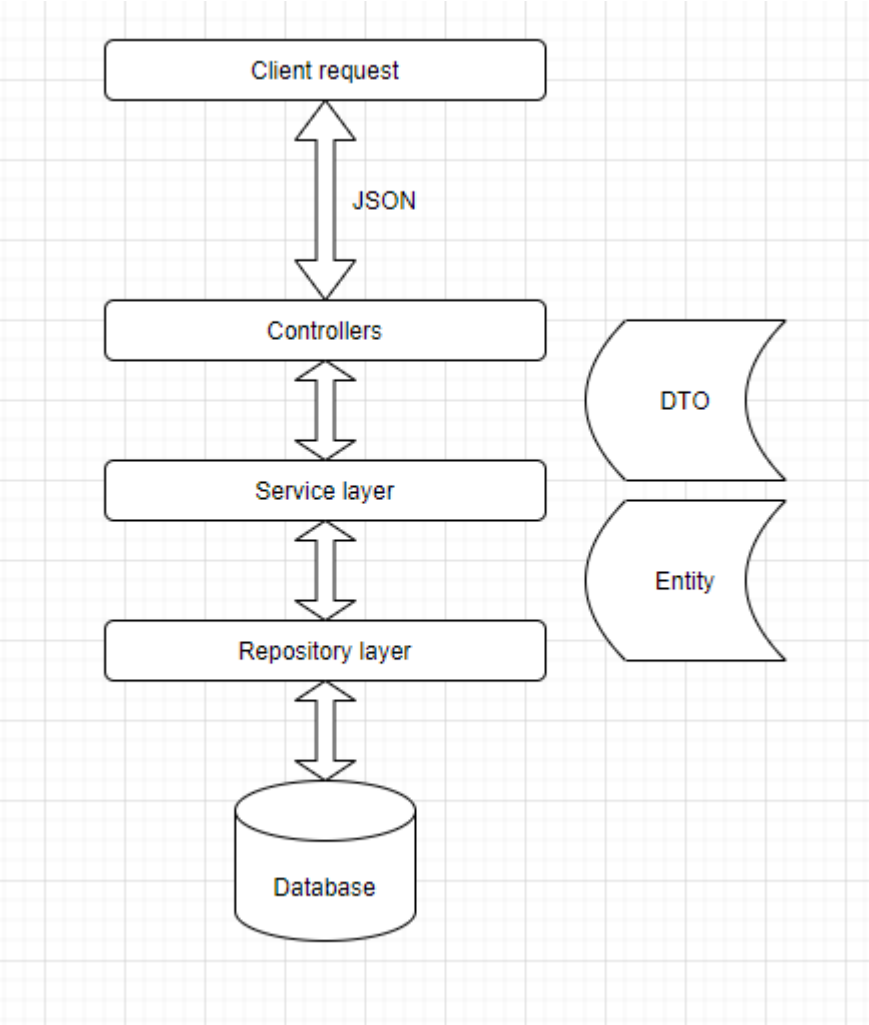

Рис. 2.20 Схема роботи системи

Отже, за схемою, розроблювана система має працювати наступним чином:

1. Клієнтський запит, що був сформований на фронтенді у форматі джсон надходить на шар контролерів. За допомогою класу DispatcherServlet, Spring направляє запит на відповідний метод відповідного класу-контролеру.

2. Після надходження на відповідний метод, керування приймає шар сервісу, куди з контролера перенаправляються тіло запиту та допоміжні змінні.

В контролерах також відбувається первісна валідація надходжених даних.

Для цього нам необхідно у відповідному методі контролера поряд із класом тіла запиту поставити анотацію @Valid.

```
@PostMapping
public ResponseEntity<UserDto> createUser(@RequestBody @Valid CreateUserDto userDto) {
    return new ResponseEntity(userService.create(userDto), HttpStatus.CREATED);
}
```

Рис. 2.21 Приклад використання анотації для валідації

А у відповідному класі прописати правила валідації:

```
public class CreateUserDto extends AbstractBaseDto {

    @NotEmpty
    String firstName;

    @NotEmpty
    String lastName;

    @NotEmpty
    String patronymicName;

    @Pattern(regexp = "^(?=.*[a-z])(?=.*[A-Z])(?=.*\\d)[a-zA-Z\\d]{6,24}$")
    @NotEmpty
    String password;

    @Email
    @NotEmpty
    String email;

    @Size(max = 12)
    @NotEmpty
    String login;

    @JsonFormat(shape = JsonFormat.Shape.STRING, pattern = "yyyy-MM-dd")
    Date birthDate;

    String userType;

}
```

Рис. 2.22 Приклад правил валідації

3. В свою чергу сервіси взаємодіють із шаром-репозиторіїв, формуючи різноманітні SQL-запити відповідно до бізнес логіки.

4. Шар репозиторіїв містить класи-моделі, що є проекціями таблиць у базі даних, та власне репозиторії, за допомогою яких отримується доступ до БД.

5. Також варто не забувати про конфігурації та рівень безпеки. Кожний запит має містити відповідний токен доступу, що зберігає інформацію про користувача та надає тому доступ до ендпоінтів в залежності від його ролі.

База даних:

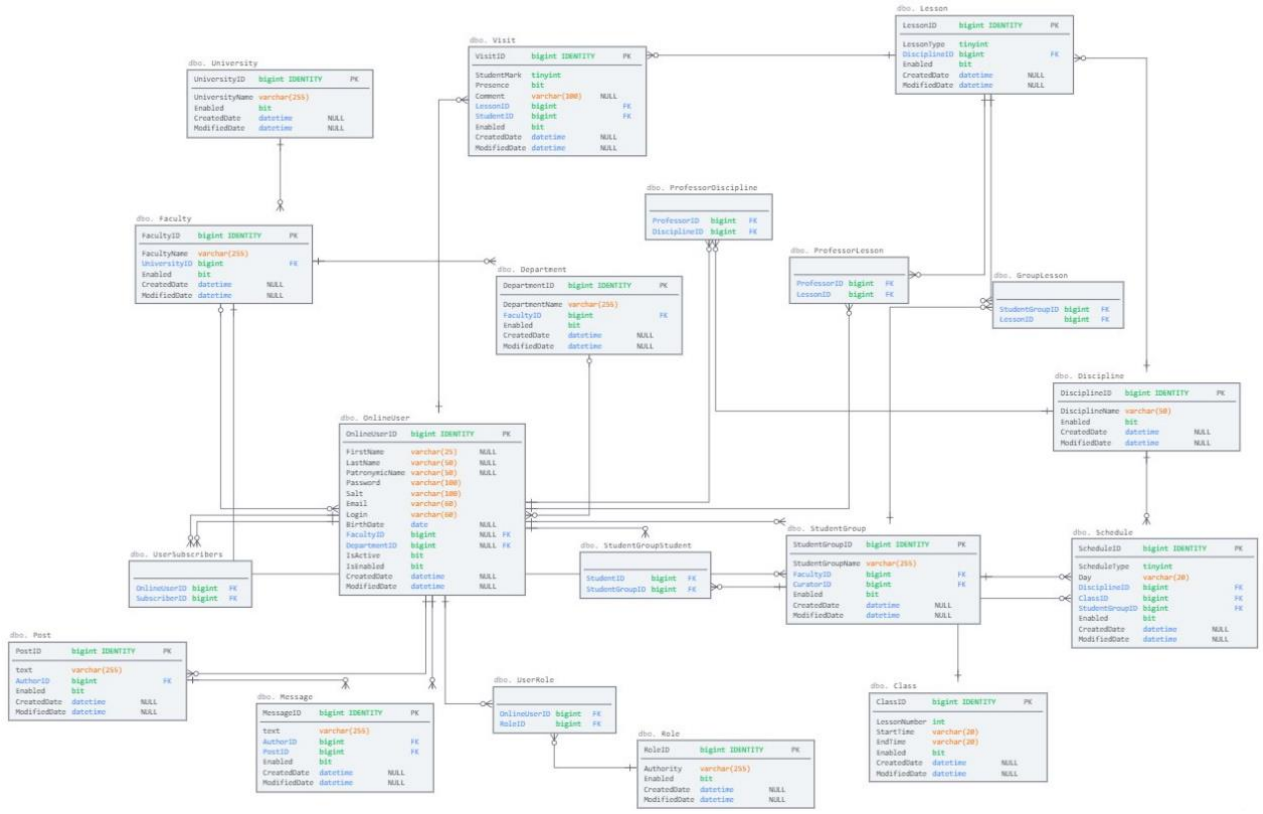

Рис. 2.23 Схема бази даних

Основні таблиці:

Таблиця 2.8

Структура об’єкту OnlineUser

Назва колонки	Тип	Призначення
id	bigint	Унікальний ідентифікатор користувача

user_type	varchar(discriminator column)	Тип користувача
created_date	datetime	Дата створення запису
modified_date	datetime	Дата редагування запису
enabled	boolean	Показує стан користувача
active	boolean	Показує чи активний користувач
first_name	varchar	Ім’я користувача
last_name	varchar	Прізвище користувача
patronymic_name	varchar	По-батькові користувача
login	varchar	Логін користувача, має бути унікальним
email	varchar	Пошта користувача, має бути унікальною
password	varchar	Пароль користувача в зашифрованому вигляді
salt	varchar	Допоміжна строка для шифрування паролю
birth_date	date	Дата народження користувача
department_id	bigint	Ідентифікатор кафедри, до якої належить користувач (якщо користувач - викладач). Зовнішній ключ.
faculty_id	bigint	Ідентифікатор факультету, до якого належить користувач (якщо користувач – викладач та не належить до жодної кафедри). Зовнішній ключ.

Таблиця 2.9

Структура об’єкту Discipline

Назва колонки	Тип	Призначення
id	bigint	Унікальний ідентифікатор дисципліни
discipline_name	varchar	Назва дисципліни, має бути унікальною
created_date	datetime	Дата створення запису
modified_date	datetime	Дата редагування запису
enabled	boolean	Показує стан дисципліни

Таблиця 2.10

Структура об’єкту Faculty

Назва колонки	Тип	Призначення
id	bigint	Унікальний ідентифікатор факультету
faculty_name	varchar	Назва факультету, має бути унікальною
created_date	datetime	Дата створення запису
modified_date	datetime	Дата редагування запису
enabled	boolean	Показує стан факультету
university_id	bigint	Ідентифікатор університету, до якого належить факультет. Зовнішній ключ

Таблиця 2.11

Структура об’єкту Department

Назва колонки	Тип	Призначення
id	bigint	Унікальний ідентифікатор кафедри
department_name	varchar	Назва кафедри, має бути унікальною
created_date	datetime	Дата створення запису
modified_date	datetime	Дата редагування запису
enabled	boolean	Показує стан кафедри
faculty_id	bigint	Ідентифікатор факультету, до якого належить кафедра. Зовнішній ключ

Таблиця 2.12

Структура об’єкту Post

Назва колонки	Тип	Призначення
id	bigint	Унікальний ідентифікатор посту
text	varchar	Текст посту
created_date	datetime	Дата створення запису
modified_date	datetime	Дата редагування запису
enabled	boolean	Показує стан посту
author_id	bigint	Ідентифікатор користувача, що є автором посту. Зовнішній ключ

Таблиця 2.13

Структура об’єкту Comment

Назва колонки	Тип	Призначення
id	bigint	Унікальний ідентифікатор коментаря
text	varchar	Текст коментаря
created_date	datetime	Дата створення запису
modified_date	datetime	Дата редагування запису
enabled	boolean	Показує стан коментаря
author_id	bigint	Ідентифікатор користувача, що є автором коментаря. Зовнішній ключ
post_id	bigint	Ідентифікатор посту, до якого був написаний коментар. Зовнішній ключ

Таблиця 2.14

Структура об’єкту StudentGroup

Назва колонки	Тип	Призначення
id	bigint	Унікальний ідентифікатор групи
group_name	varchar	Назва групи, має бути унікальною
created_date	datetime	Дата створення запису
modified_date	datetime	Дата редагування запису
enabled	boolean	Показує стан групи
faculty_id	bigint	Ідентифікатор факультету, до якого належить група. Зовнішній ключ

curator_id	bigint	Ідентифікатор користувача, що є куратором групи. Зовнішній ключ

Таблиця 2.15

Структура об’єкту Schedule

Назва колонки	Тип	Призначення
id	bigint	Унікальний ідентифікатор розкладу
created_date	datetime	Дата створення запису
modified_date	datetime	Дата редагування запису
enabled	boolean	Показує стан розкладу
day	varchar	День тижня
type	byte	Тип розкладу(1 – чисельник, 2 - знаменник)
class_id	bigint	Ідентифікатор уроку в розкладі(номер пари) . Зовнішній ключ
discipline_id	bigint	Ідентифікатор дисципліни. Зовнішній ключ
group_id	bigint	Ідентифікатор групи. Зовнішній ключ

Таблиця 2.16

Структура об’єкту Role

Назва колонки	Тип	Призначення
id	bigint	Унікальний ідентифікатор ролі

authority	varchar	Назва ролі, має бути унікальною
created_date	datetime	Дата створення запису
modified_date	datetime	Дата редагування запису
enabled	boolean	Показує стан ролі

2.6. Обґрунтування та організація вхідних та вихідних даних програми.

Вхідні дані надаються системі у форматі JSON, що формуються на фронт-енді в залежності від дій користувача. В залежності від необхідної інформації, фронт-енд надсилає відповідні запити до розроблюваної системи, котра в свою чергу, оброблює ці запити та відповідає вихідними даними у форматі JSON.

2.7. Опис розробленого програмного продукту.

2.7.1. Використані технічні засоби.

В процесі тестування та розробки ІС були використані наступні технічні засоби:

- оперативна пам’ять обсягом 12 гб;
- процесор AMD FX-8350 4.00 GHz;
- накопичувач на жорсткому диску обсягом 250 гб;
- відеоадаптер NVIDIA GeForce GTX 1060;
- клавіатура та маніпулятор типу «миш».

Вказані технічні засоби не мають бути обов’язково ідентичних характеристик для безперешкодної роботи системи.

2.7.2. Використані програмні засоби

Під час розробки даного застосунку були використані такі програмні засоби:

– IntelliJ IDEA;

– Git, GitHub.

– AWS (RDS)

– Heroku

IntelliJ IDEA

«IntelliJ IDEA – це особливе середовище програмування або інтегроване середовище розробки (IDE), багато в чому призначене для Java». Це середовище використовується спеціально для розробки програм. Воно розроблене компанією під назвою JetBrains. Воно доступне у двох виданнях: Community Edition, що має ліцензію Apache 2.0, і комерційне видання, відоме як Ultimate Edition. Що робить IntelliJ IDEA настільки відмінним від своїх аналогів, це його легкість у використанні, гнучкість та зручний дизайн.

GIT

Git – це приклад системи управління розподіленими версіями (DVCS), яка зазвичай використовується для розробки відкритого комерційного та некомерційного програмного забезпечення. DVCS дозволяють отримати повний доступ до кожного файлу, гілки та ітерації проекту та дозволяє кожному користувачеві отримати доступ до повної та самодостатньої історії всіх змін. На відміну від популярних колись централізованих систем управління версіями, «DVCS типу Git не потребує постійного підключення до центрального сховища.

GitHub – це платформа для розміщення коду для контролю версій та співпраці. Він дозволяє спільно працювати над проектами з будь-якого місця та зберігати версії проекту на віддаленому сервері.

AWS (RDS)

Amazon Relational Database Service (Amazon RDS) – це хмарний сервіс, що дозволяє просто налаштовувати, використовувати і масштабувати реляційні бази даних. Сервіс забезпечує автоматизацію трудомістких завдань адміністрування,

таких як виділення апаратного забезпечення, налаштування бази даних, установка виправлень і резервне копіювання. Це дозволяє зосередити увагу на додатках, щоб забезпечити для них високу продуктивність, доступність, безпеку і сумісність.

Amazon RDS доступний у вигляді інстанси бази даних декількох типів: оптимізовані для роботи з пам'яттю, для високої продуктивності або виконання операцій введення-виведення - і пропонує на вибір шість відомих ядер баз даних, в тому числі Amazon Aurora, PostgreSQL, MySQL, MariaDB, Oracle Database і SQL Server. За допомогою сервісу AWS Database Migration Service можна просто перенести або реплікувати існуючі бази даних в Amazon RDS.

Heroku

Heroku - це хмарна платформа на основі контейнерів. Розробники використовують Heroku для розгортання, управління та масштабування сучасних програм. Варто відзначити, що ця платформа доволі гнучка та інтуїтивно проста у використанні, пропонуючи розробникам найпростіший шлях до випуску своїх програм на ринок.

Heroku надає повний контроль над застосунком, що дає розробникам свободу зосередитися на розробці функціоналу, не відволікаючи на підтримку серверів, обладнання та інфраструктури.

2.7.3. Виклик та завантаження програми

Для роботи з розробленим веб-додатком потрібно лише звернутися за адресою https://socialnet-thesis.herokuapp.com/ (підтримується хмарним сервісом Heroku та не потребує додаткових дій або ПЗ, за виключенням засобу звернення до інтернету – браузера, вбудованих сервісів Windows тощо).

2.7.4. Опис інтерфейсу користувача

В системі використовується візуалізація розробленого API – OpenAPI Swagger.

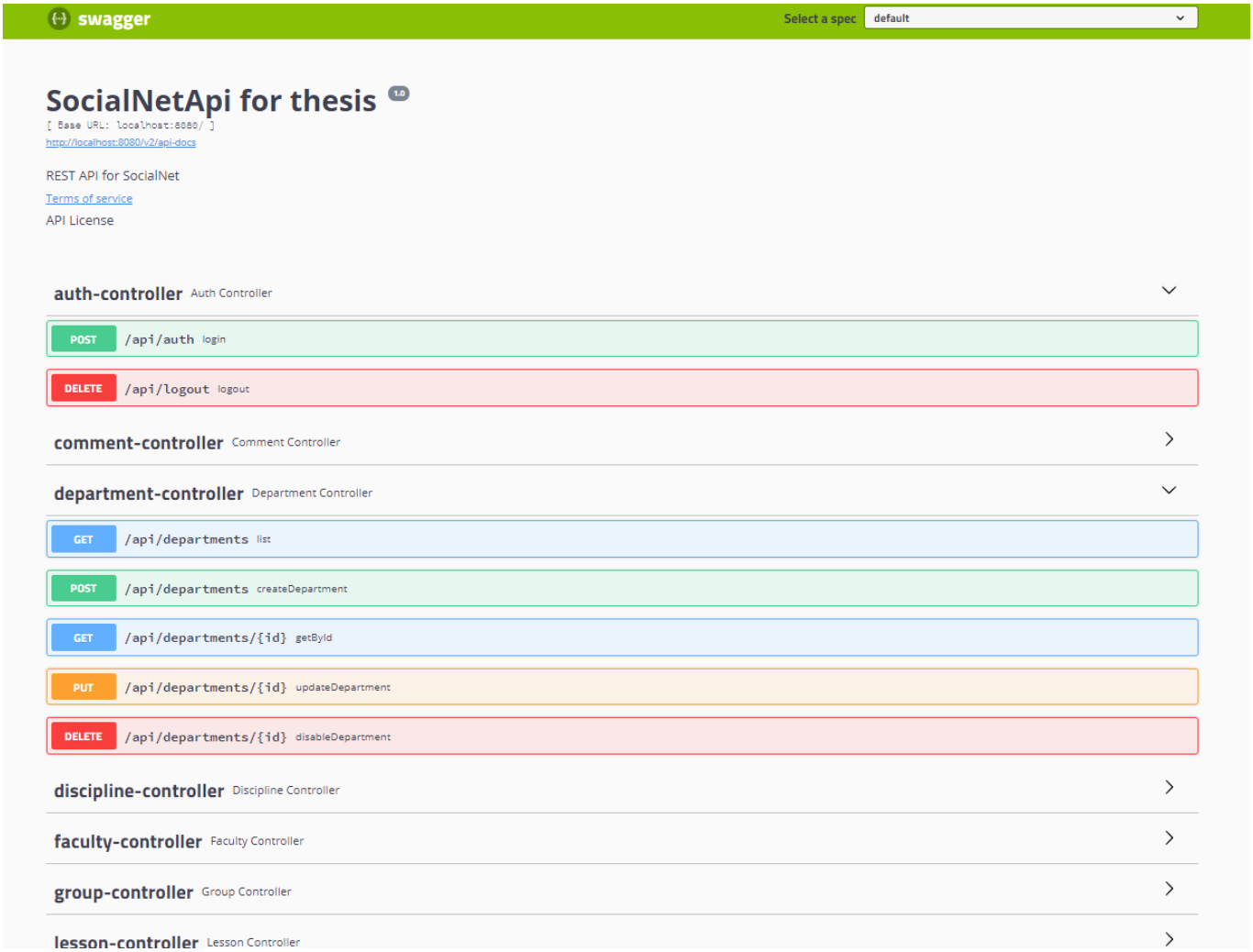

Рис. 2.24 OpenAPI

За допомогою цієї сторінки можна дивитись існуючі ендпоїнти та надсилати запити.

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Розрахунок трудомісткості та вартості розробки програмного продукту

Початкові дані:

1. передбачуване число операторів програми – 3386;

2. коефіцієнт складності програми – 1,3;

3. коефіцієнт корекції програми в ході її розробки – 0,07;

4. годинна заробітна плата програміста – 150 грн/год;

5. коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі – 1,3;

Середня зарплата програміста у годину була вирахувана виходячи з даних «Української спільноти програмістів (DOU)». Середньоукраїнська заробітна плата програміста, який пише програми на мові програмування Java і з досвідом роботи близько року дорівнює коливається від 800 до 1000 американських доларів в місяць[12], тож очікувана заробітна плата має бути в цих рамках. При поточному курсі валют НБУ станом на початок червня 2021 року один американський долар дорівнює 27,17 грн[13], тому середня зарплата в гривнях дорівнює 26400 грн (971 доларів США).

6. коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності – 1,2;

7. вартість машино-години ЕОМ – 23,16 грн/год.

Комп'ютер споживає приблизно 250 Ватт / год, що дорівнює 0,25 кВт / год. Вартість одного кВт на годину від постачальника Yasno на стан середини 2021 року дорівнює 1,68 грн в незалежності від обсягом споживання. Отже вартість електроенергії споживаної комп'ютером за годину дорівнює 0,42 грн. Ціна довгострокової оренди ноутбука дорівнює 4000 грн на місяць[14], з чого можна

вирахувати, що ціна оренди на годину буде дорівнювати 22,72 грн. Остаточна вартість машино-години ЕОМ дорівнює 23,16 грн за годину[15].

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки ПЗ може бути розрахована на основі системи моделей з різною точністю оцінки.

Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_o + t_u + t_a + t_n + t_{отл} + t_д \text{, людино-годин,} \quad (3.1)$$

де t_o – витрати праці на підготовку й опис поставленої задачі (приймається 50);

t_u – витрати праці на дослідження алгоритму рішення задачі;

t_a – витрати праці на розробку блок-схеми алгоритму;

t_n – витрати праці на програмування по готовій блок-схемі;

$t_{отл}$ – витрати праці на налагодження програми на ЕОМ;

$t_д$ – витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у ПЗ, яке розробляється.

Умовне число операторів (підпрограм):

$$Q = q \cdot C \cdot (1 + p), \text{ де} \quad (3.2)$$

q – передбачуване число операторів;

C – коефіцієнт складності програми;

p – коефіцієнт кореляції програми в ході її розробки.

$Q = 3386 \cdot 1{,}3 \cdot (1 + 0{,}07) = 4709{,}926;$

Витрати праці на вивчення опису задачі t_u визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{Q \cdot B}{(75 \ldots 85) \cdot K}, \text{ людино-годин,} \quad (3.3)$$

де B – коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі;

K – коефіцієнт кваліфікації програміста, обумовлений стажем роботи з даної спеціальності;

$t_u = \frac{4709{,}926 \cdot 1{,}3}{85 \cdot 1{,}2} = 60{,}02$, людино-годин.

Витрати праці на розробку алгоритму рішення задачі:

$$t_a = \frac{Q}{(20...25) \cdot K}; \quad (3.4)$$

$t_a = \frac{4709{,}926}{20 \cdot 1{,}2} = 196{,}24$, людино-годин.

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20...25) \cdot K}; \quad (3.5)$$

$t_n = \frac{4709{,}926}{25 \cdot 1{,}2} = 157$, людино-годин.

Витрати праці на налагодження програми на ЕОМ:

– за умови автономного налагодження одного завдання:

$$t_{\text{отл}} = \frac{Q}{(4...5) \cdot K}; \quad (3.6)$$

$t_n = \frac{4709{,}926}{5 \cdot 1{,}2} = 785$, людино-годин,

– за умови комплексного налагодження завдання:

$$t_{\text{отл}}^{\text{к}} = 1{,}2 \cdot t_{\text{отл}}; \quad (3.7)$$

$t_{\text{отл}}^{\text{к}} = 1{,}2 \cdot 810{,}108 = 942$, людино-годин

Витрати праці на підготовку документації:

$$t_\partial = t_{\partial p} + t_{\partial o}; \quad (3.8)$$

де $t_{\partial p}$ – трудомісткість підготовки матеріалів і рукопису

$$t_\partial = \frac{Q}{(15...20) \cdot K}; \quad (3.9)$$

$t_\partial = \frac{4709{,}926}{20 \cdot 1{,}2} = 196{,}24$, людино-годин.

$t_{\partial o}$– трудомісткість редагування, печатки й оформлення документації

$$t_{\partial o} = 0{,}75 \cdot t_{\partial \text{р}}; \quad (3.10)$$

$t_{\partial o} = 0{,}75 \cdot 196{,}24 = 147{,}18$, людино-годин.

$t_\partial = 196{,}24 + 147{,}18 = 343{,}42$, людино-годин.

Отримаємо трудомісткість розробки програмного забезпечення:

t = 50 + 60,02 + 196,24 + 157 + 942 + 343,42= 1748,66, людино-годин.

У результаті ми розрахували, що в загальній складності необхідно 1748,66 людино-годин для розробки даного програмного забезпечення.

3.2. Розрахунок витрат на створення програми

Витрати на створення ПЗ Кпо включають витрати на заробітну плату виконавця програми Зз/п і витрат машинного часу, необхідного на налагодження програми на ЕОМ.

$$К_{ПО} = З_{ЗП} + З_{МВ}, \text{ грн,} \quad (3.11)$$

де $З_{ЗП}$ – заробітна плата виконавців, яка визначається за формулою:

$$З_{ЗП} = t \cdot С_{ПР}, \text{ грн,} \quad (3.12)$$

де t – загальна трудомісткість, людино-годин;

$С_{ПР}$ – середня годинна заробітна плата програміста, грн/година

$$З_{ЗП} = 1748{,}66 \cdot 150 = 262299, \text{ грн.}$$

$З_{МВ}$ – Вартість машинного часу, необхідного для налагодження програми на ЕОМ:

$$З_{МВ} = t_{отл} \cdot С_{М}, \text{ грн,} \quad (3.13)$$

де $t_{отл}$ – трудомісткість налагодження програми на ЕОМ, год.

$С_{МЧ}$ – вартість машино-години ЕОМ, грн/год.

$$З_{МВ} = 785 \cdot 23{,}16 = 18180{,}6 \text{ грн.}$$

$$К_{ПО} = 262299 + 18180{,}6 = 280479{,}6 \text{ грн.}$$

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k \cdot F_p}, \text{ мес.} \quad (3.14)$$

де B_k- число виконавців;

$F_р$ – місячний фонд робочого часу (при 40 годинному робочому тижні Fр=176 годин).

$$T = \frac{1748{,}66}{1 \cdot 176} = 9{,}9 \text{ міс.}$$

Висновки. На розробку даного програмного забезпечення піде 1748,66 людино-годин. Тобто, ймовірна очікувана тривалість розробки складатиме 9,9 місяці при стандартному 40-годинному робочому тижні і 176-годинному

робочому місяці. Очікувані витрати на створення програмного забезпечення складатимуть 280479,6 грн.

Список використаних джерел

1. Мартін Калін. Java Web Services: Up and Running, 2009. 320 с.
2. Крейг Уолс. Спрінг у Дії, Шосте Видання, 2021. 520 с.
3. Алєксєєва І.Ю. Людське знання і його комп'ютерний образ. М., 1993. 267 с.
4. Юліана Косміна. Pivotal Certified Professional Core Spring 5 Developer Exam: A Study Guide Using, 2020, 609 с.
5. Бріллюен Л. Наука і теорія інформації. М., 1959 – 97 с.
6. Брайан Соліс. Роль сучасних соціальних мереж в соціумі та політичних технологіях. Директ-Медиа 2012. 8с.
7. Ших К. Ера Facebook. Як використовувати можливості соціальних мереж для розвитку бізнесу / Клара Ших. – М.: Ман, 2011. – 304 с.
8. JSON Web Token Introductio. URL: https://jwt.io/introduction
9. СОЦІАЛЬНІ МЕРЕЖІ ЯК ЧИННИК ІНФОРМАЦІЙНОЇ БЕЗПЕКИ. URL: http://nbuviap.gov.ua/images/sozinfo/2014/s-net21.pdf
10. Соціальні мережі та електронні платформи для науковців. URL: https://osvita.ua/vnz/76103/
11. Рой Томас Філдінг. Architectural Styles and the Design of Network-based Software Architectures URL: https://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf
12. Веб-сайт «Українська спільнота програмістів», URL: https://jobs.dou.ua/salaries/#period=dec2020&city=all&title=Junior%20Software%20Engineer&language=Java&spec=&exp1=0&exp2=2
13. Веб-сайт НБУ, URL: https://bank.gov.ua/ua/markets/exchangerate-chart?cn%5B%5D=USD
14. Веб-сайт компанії орендатора комп'ютерів «ChipChip», URL: https://komputers.com.ua/arenda-kompyutera/
15. Веб-сайт постачальника електроенергії Yasno, URL: https://yasno.com.ua/b2c-tariffs
16. Дірк Кьоніг. Груві у Дії, 2007, 48 с.

17. Кішорі Шаран. Особливості мови програмування Java. 2018 р. 236 с.
18. Документація LiquiBase. URL: https://docs.liquibase.com/home.html

ДОДАТОК А

КОД ПРОГРАМИ

SocialNetApplication.java

```
@SpringBootApplication(scanBasePackages = "ua.ntudp.isvyry.socialnet")
public class SocialNetApplication {

   public static void main(String[] args) {
      SpringApplication.run(SocialNetApplication.class, args);
   }

}
```

JwtFilter.java

```
@RequiredArgsConstructor
@Component
public class JwtFilter extends GenericFilterBean {

   public static final String AUTHORIZATION = "Authorization";

   private final JwtProvider jwtProvider;

   @Qualifier("socialNetUserDetailsService")
   private final UserDetailsService userDetailsService;

   @Override
   public void doFilter(ServletRequest servletRequest, ServletResponse servletResponse, FilterChain filterChain) throws IOException, ServletException {
      String token = getTokenFromRequest((HttpServletRequest) servletRequest);
      if (token != null && jwtProvider.validateToken(token)) {
         String userLogin = jwtProvider.getLoginFromToken(token);
         SocialNetUserDetails userDetails = (SocialNetUserDetails) userDetailsService.loadUserByUsername(userLogin);
         UsernamePasswordAuthenticationToken auth = new UsernamePasswordAuthenticationToken(userDetails, null, userDetails.getAuthorities());
```

```
            SecurityContextHolder.getContext().setAuthentication(auth);
        }
        filterChain.doFilter(servletRequest, servletResponse);
    }

    private String getTokenFromRequest(HttpServletRequest request) {
        String bearer = request.getHeader(AUTHORIZATION);
        if (hasText(bearer) && bearer.startsWith("Bearer ")) {
            return bearer.substring(7);
        }
        return null;
    }
}
```

SecurityConfig.java

```
@Configuration
@EnableWebSecurity
@RequiredArgsConstructor
@EnableGlobalMethodSecurity(prePostEnabled = true)
public class SecurityConfig extends WebSecurityConfigurerAdapter {

    @Value("${security.allowed-endpoints}")
    private String[] allowedEndpoints;

    private final JwtFilter jwtFilter;

    @Qualifier("socialNetUserDetailsService")
    private final UserDetailsService userDetailsService;

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http
                .httpBasic().disable()
                .csrf().disable()
```

```
.sessionManagement().sessionCreationPolicy(SessionCreationPolicy.STATELESS)
            .and()
            .authorizeRequests()
            .antMatchers("/api/auth", "/api/auth/*").permitAll()
            .antMatchers(allowedEndpoints).permitAll()
            // all other requests need to be authenticated
            .anyRequest().authenticated()
            .and().logout()
            //.logoutRequestMatcher(new AntPathRequestMatcher("/api/auth"))
            .deleteCookies("Authorization")
            .invalidateHttpSession(true).permitAll()
            .and()
            .addFilterBefore(jwtFilter, UsernamePasswordAuthenticationFilter.class);

    }

    /** configure AuthenticationManager
     * so that it knows from where to load user for matching credentials, use PasswordEncoder
     */
    @Autowired
    public void configureGlobal(AuthenticationManagerBuilder auth) throws Exception {
        auth.userDetailsService(userDetailsService).passwordEncoder(passwordEncoder());
    }

    @Bean
    public PasswordEncoder passwordEncoder() {
        return NoOpPasswordEncoder.getInstance();
    }

    @Bean
    @Override
    public AuthenticationManager authenticationManagerBean() throws Exception {
        return super.authenticationManagerBean();
    }
```

}

SocialNetUserDetails.java

```
public class SocialNetUserDetails implements UserDetails {

    private final User user;

    private final Collection<Role> userRoles;

    public SocialNetUserDetails(User user, Collection<Role> userRoles) {
        this.user = user;
        this.userRoles = userRoles;
    }

    @Override
    public Collection<? extends GrantedAuthority> getAuthorities() {
        return userRoles;
    }

    public Long getId() {
        return user.getId();
    }

    @Override
    public String getPassword() {
        return user.getPassword();
    }

    @Override
    public String getUsername() {
        return user.getLogin();
    }

    @Override
    public boolean isAccountNonExpired() {
```

```
        return true;
    }

    @Override
    public boolean isAccountNonLocked() {
        return true;
    }

    @Override
    public boolean isCredentialsNonExpired() {
        return true;
    }

    @Override
    public boolean isEnabled() {
        return user.getEnabled();
    }
}
```

SocialNetUserDetailsService.java

```
@Component
@RequiredArgsConstructor
public class SocialNetUserDetailsService implements UserDetailsService {

    private final UserRepository userRepository;

    @Override
    @Transactional(readOnly=true)
    public UserDetails loadUserByUsername(String login) throws
UsernameNotFoundException {
        User user = userRepository.findByLogin(login)
                .orElseThrow(() -> new EntityNotFoundException("User with given login doesn't
exist: " + login));
        Set<Role> userRoles = user.getUserRoles();
        return new SocialNetUserDetails(user, userRoles);
```

```
    }
}
```

BaseEntity.java

```
Data
@MappedSuperclass
public abstract class BaseEntity {

    LocalDateTime createdDate;

    LocalDateTime modifiedDate;

    Boolean enabled = true;

}
```

User.java

```
@Entity
@Table(name = "OnlineUser")
@Inheritance(strategy = InheritanceType.SINGLE_TABLE)
@DiscriminatorColumn(name="UserType",
        discriminatorType = DiscriminatorType.STRING)
@Data
public class User extends BaseEntity {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    protected Long id;

    String firstName;

    String lastName;

    String patronymicName;

    String password;
```

```
String salt;

String login;

String email;

Date birthDate;

boolean active = true;

boolean enabled = true;

@OneToMany(mappedBy = "author")
Set<Post> posts = new HashSet<>();

@ManyToMany
@JoinTable(name = "usersubscribers",
      joinColumns = @JoinColumn(name = "SubscriptionID"),
      inverseJoinColumns = @JoinColumn(name = "SubscriberID"))
Set<User> subscriptions = new HashSet<>();

@ManyToMany
@JoinTable(name = "usersubscribers",
      joinColumns = @JoinColumn(name = "SubscriberID"),
      inverseJoinColumns = @JoinColumn(name = "SubscriptionID"))
Set<User> subscribers = new HashSet<>();

@OneToMany(mappedBy = "author")
Set<Comment> comments = new HashSet<>();

@ManyToMany(mappedBy = "users")
Set<Role> userRoles = new HashSet<>();

@Transient
public String getDecriminatorValue() {
```

```
        return this.getClass().getAnnotation(DiscriminatorValue.class).value();
    }
}
```

Professor.java

```
@Entity
@DiscriminatorValue("Professor")
@Getter
@Setter
@NoArgsConstructor
public class Professor extends User {

    @ManyToOne
    Faculty faculty;

    @OneToMany(mappedBy = "curator")
    Set<Group> groups = new HashSet<>();

    @ManyToOne
    Department department;

    @ManyToMany
    @JoinTable(name = "lessonteachers",
            joinColumns = @JoinColumn(name = "TeacherID"),
            inverseJoinColumns = @JoinColumn(name = "LessonID"))
    private Set<Lesson> lessons = new HashSet<>();

    @ManyToMany
    @JoinTable(name = "disciplineteachers",
            joinColumns = @JoinColumn(name = "TeacherID"),
            inverseJoinColumns = @JoinColumn(name = "DisciplineID"))
    Set<Discipline> disciplines = new HashSet<>();

    public Professor(User user) {
        this.id = user.getId();
```

```
        this.firstName = user.getFirstName();
        this.lastName = user.getLastName();
        this.patronymicName = user.getPatronymicName();
        this.password = user.getPassword();
        this.salt = user.getSalt();
        this.email = user.getEmail();
        this.login = user.getLogin();
        this.comments = user.getComments();
        this.posts = user.getPosts();
        this.subscribers = user.subscribers;
        this.subscriptions = user.subscriptions;
        this.userRoles = user.getUserRoles();
        this.birthDate = user.getBirthDate();
        this.active = user.active;
    }
}
```

Student.java

```
@Entity
@DiscriminatorValue("Student")
@NoArgsConstructor
@Getter
@Setter
public class Student extends User {

    @ManyToMany
    @JoinTable(name = "usergroup",
            joinColumns = @JoinColumn(name = "OnlineUserID"),
            inverseJoinColumns = @JoinColumn(name = "GroupID"))
    Set<Group> groups = new HashSet<>();

    @OneToMany(mappedBy = "student")
    Set<Visit> visits = new HashSet<>();

    public Student(User user) {
```

```
        this.id = user.getId();
        this.firstName = user.getFirstName();
        this.lastName = user.getLastName();
        this.patronymicName = user.getPatronymicName();
        this.password = user.getPassword();
        this.salt = user.getSalt();
        this.email = user.getEmail();
        this.login = user.getLogin();
        this.comments = user.getComments();
        this.posts = user.getPosts();
        this.subscribers = user.subscribers;
        this.subscriptions = user.subscriptions;
        this.userRoles = user.getUserRoles();
        this.birthDate = user.getBirthDate();
        this.active = user.active;
    }
}
```

Post.java

```
@Entity
@Getter
@Setter
public class Post extends BaseEntity {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "PostID")
    private Long id;

    String text;

    LocalDateTime createdDate;

    LocalDateTime modifiedDate;

    @ManyToOne
```

```
    User author;

    @OneToMany(mappedBy = "post")
    Set<Comment> comments = new HashSet<>();
}
```

Comment.java

```
@Entity
@Getter
@Setter
public class Comment extends BaseEntity {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @ManyToOne
    private User author;

    private String text;

    @ManyToOne
    private Post post;

}
```

Role.java

```
@Entity
@Getter
@Setter
public class Role extends BaseEntity implements GrantedAuthority {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "RoleID")
    private Long id;
```

```
    private String authority;

    @ManyToMany
    @JoinTable(name = "userrole",
            joinColumns = @JoinColumn(name = "RoleID"),
            inverseJoinColumns = @JoinColumn(name = "OnlineUserID"))
    Set<User> users;

    public String getAuthority() {
        return authority;
    }
}
```

Class.java

```
@Entity
@Getter
@Setter
public class Class extends BaseEntity {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private Integer lessonNumber;

    private String startTime;

    private String endTime;

    @OneToMany(mappedBy = "aClass")
    private Set<Schedule> schedules = new HashSet<>();

    @OneToMany(mappedBy = "lessonClass")
    private Set<Lesson> lessons = new HashSet<>();
}
```

Schedule.java

```
@Entity
@Getter
@Setter
public class Schedule extends BaseEntity {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    Long id;

    Byte type;

    @ManyToOne
    Discipline discipline;

    @ManyToOne
    Class aClass;

    @Enumerated(EnumType.STRING)
    DayOfWeek day;

    @ManyToOne
    Group group;
}
```

Discipline.java

```
@Entity
@Getter
@Setter
public class Discipline extends BaseEntity {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
```

```
    @Column(name = "DisciplineID")
    private Long id;

    @ManyToMany(mappedBy = "disciplines")
    private Set<Professor> teachers;

    private String disciplineName;

    @OneToMany(mappedBy = "discipline")
    private Set<Lesson> lessons = new HashSet<>();

    @OneToMany(mappedBy = "discipline")
    private Set<Schedule> schedules = new HashSet<>();
}
```

Group.java

```
@Entity
@Getter
@Setter
@Table(name = "StudentGroup")
public class Group extends BaseEntity {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "GroupID")
    Long id;

    String groupName;

    @ManyToOne
    Faculty faculty;

    @ManyToMany(mappedBy = "groups")
    Set<Student> students = new HashSet<>();

    @ManyToMany
```

```
    @JoinTable(name = "grouplesson",
            joinColumns = @JoinColumn(name = "GroupID"),
            inverseJoinColumns = @JoinColumn(name = "LessonID"))
    Set<Lesson> lessons = new HashSet<>();

    @ManyToOne
    Professor curator;

    @OneToMany(mappedBy = "group")
    Set<Schedule> groupSchedule = new HashSet<>();
}
```

CustomScheduleRepository.java

```
public interface CustomScheduleRepository {

    Map<String, Map<DayOfWeek, List<ScheduleSlimTransformer>>>
getScheduleByStudent(Student student, Integer scheduleType);

    Map<DayOfWeek, List<ScheduleSlimTransformer>> getScheduleByGroup(Group group,
Integer scheduleType);
}
```

CustomScheduleRepositoryImpl.groovy

```
@Repository
@Transactional(readOnly = true)
class ScheduleRepositoryImpl implements CustomScheduleRepository {

    @PersistenceContext
    EntityManager manager

    SessionFactory sessionFactory;

    @Autowired
    ScheduleRepositoryImpl(EntityManagerFactory factory) {
        this.sessionFactory =
```

```
        factory.unwrap(SessionFactory.class);
    }

    @Override
    Map<String, Map<DayOfWeek, List<ScheduleSlimTransformer>>>
getScheduleByStudent(Student student, Integer scheduleType) {
        Map<String, Map<DayOfWeek, List<ScheduleSlimTransformer>>> resultMap =
            student.getGroups().collectEntries {[ (it.getGroupName()) :
getScheduleByGroup(it, scheduleType) ] }
        resultMap
    }

    @Override
    Map<DayOfWeek, List<ScheduleSlimTransformer>> getScheduleByGroup(Group group,
Integer scheduleType) {

        List<SocialNetScheduleTransformer> list =
sessionFactory.currentSession.createSQLQuery("""select sc.day as day, d.discipline_name
as discipline, c.lesson_number as lessonNum, c.start_time as startTime, c.end_time as
endTime from Schedule as sc
            left join student_group as sg
            on sg.groupid=sc.group_groupid
            left join discipline as d
            on d.disciplineid = sc.discipline_disciplineid
                                join class as c on c.id= sc.a_class_id
            and sg.group_name = :groupName
            and sc.type = :scheduleType
            order by sc.day""")
            .setString("groupName", group.getGroupName())
            .setInteger("scheduleType", scheduleType)
            .setResultTransformer(ScheduleTransformer.INSTANCE)
            .list()

        Map map = list.groupBy { SocialNetScheduleTransformer s -> s.day }
```

```
        Map newMap = map.collectEntries { Map.Entry<DayOfWeek, List<SocialNetScheduleTransformer>> entry ->
            [ (entry.getKey()) : entry.getValue().collect { s ->
                new ScheduleSlimTransformer(s.disciplineName, s.lessonNumber, s.startTime, s.endTime) } ]
        }
        newMap
    }

    private static class ScheduleTransformer implements ResultTransformer {

        static final ScheduleTransformer INSTANCE = new ScheduleTransformer();

        @Override
        Object transformTuple(Object[] tuple, String[] strings) {
            return new SocialNetScheduleTransformer(DayOfWeek.valueOf(tuple[0]), tuple[1], (Integer) tuple[2], tuple[3], tuple[4])
        }

        @Override
        List transformList(List list) {
            return list
        }
    }
}
```

ScheduleRepository.java

```
@Component
public interface ScheduleRepository extends JpaRepository<Schedule, Long>, CustomScheduleRepository {

}
```

ScheduleService.java

```
@Service
@RequiredArgsConstructor
```

```
public class ScheduleService extends AbstractCommonService<Schedule> {

    private final ScheduleRepository scheduleRepository;

    private final GroupRepository groupRepository;

    private final StudentRepository studentRepository;

    private final ClassRepository classRepository;

    private final DisciplineRepository disciplineRepository;

    @Transactional
    public Map<Byte, Map<DayOfWeek, List<ScheduleSlimTransformer>>> save(CreateScheduleDto dto) {
        return resolveAndCreateScheduleDto(dto);
    }

    @Transactional(readOnly = true)
    public Map<DayOfWeek, List<ScheduleSlimTransformer>> getScheduleByGroup(Long id, Integer scheduleType) {
        Group group = groupRepository.findById(id)
                .orElseThrow(() -> new EntityNotFoundException("Cannot find group with given id: " + id));
        Map<DayOfWeek, List<ScheduleSlimTransformer>> map = scheduleRepository.getScheduleByGroup(group, scheduleType);
        return map;
    }

    @Transactional(readOnly = true)
    public Map<String, Map<DayOfWeek, List<ScheduleSlimTransformer>>> getScheduleByStudent(Long id, Integer scheduleType) {
        Student student = studentRepository.findById(id)
                .orElseThrow(() -> new EntityNotFoundException("Cannot find user with given id: " + id));
```

```
        Map<String, Map<DayOfWeek, List<ScheduleSlimTransformer>>> map =
scheduleRepository.getScheduleByStudent(student, scheduleType);
        return map;
    }

    private Map<Byte, Map<DayOfWeek, List<ScheduleSlimTransformer>>>
resolveAndCreateScheduleDto(CreateScheduleDto scheduleMapDto) {
        Map<Byte, InnerCreateScheduleDtoMap> requestScheduleMap =
scheduleMapDto.getScheduleMap();
        Map<Byte, Map<DayOfWeek, List<ScheduleSlimTransformer>>>
createdScheduleMap = new HashMap<>();

        requestScheduleMap.forEach( (key, value) -> {
            Map<String, List<ScheduleElement>> map = value.getScheduleByDayMap();
            map.forEach( (k, v) -> {
                createdScheduleMap.put(key, createSchedule(v, DayOfWeek.valueOf(k), key));
            });
        });
        return createdScheduleMap;
    }

    private Map<DayOfWeek, List<ScheduleSlimTransformer>>
createSchedule(List<ScheduleElement> scheduleElements, DayOfWeek day, Byte
scheduleType) {
        Map<DayOfWeek, List<ScheduleSlimTransformer>> resultMap = new HashMap<>();
        List<ScheduleSlimTransformer> list = resultMap.getOrDefault(day, new
ArrayList<>());
        scheduleElements.forEach(it -> {
            Schedule schedule = new Schedule();
            Class aClass = classRepository.findByLessonNumber(it.getClassNumber())
                    .orElseThrow(() -> new EntityNotFoundException("Cannot find class with given
number: " + it.getClassNumber()));
            schedule.setAClass(aClass);

schedule.setDiscipline(disciplineRepository.findByDisciplineName(it.getDisciplineName())
```

```
                .orElseThrow(() -> new EntityNotFoundException("Cannot find discipline with given name: " + it.getDisciplineName())));
            schedule.setGroup(groupRepository.findByGroupName(it.getGroupName())
                .orElseThrow(() -> new EntityNotFoundException("Cannot find group with given name: " + it.getGroupName())));
            schedule.setDay(day);
            schedule.setType(scheduleType);
            setCreatedDate(schedule);
            scheduleRepository.save(schedule);
            list.add(new ScheduleSlimTransformer(it.getDisciplineName(), aClass.getLessonNumber(), aClass.getStartTime(), aClass.getEndTime()));
            resultMap.put(day, list);
        });
        return resultMap;
    }
}
```

ScheduleController.java

```
@RestController
@RequestMapping("/api/schedules")
@RequiredArgsConstructor
public class ScheduleController {

    private final ScheduleService scheduleService;

    @PostMapping
    public ResponseEntity<Map> createSchedule(@RequestBody CreateScheduleDto scheduleDto) {
        return new ResponseEntity(scheduleService.save(scheduleDto), HttpStatus.CREATED);
    }

    @GetMapping("/groups/{groupId}")
    public ResponseEntity getScheduleByGroup(@PathVariable Long groupId, @RequestParam("scheduleType") Integer scheduleType) {
```

```
        return new ResponseEntity(scheduleService.getScheduleByGroup(groupId,
scheduleType), HttpStatus.OK);
    }

    @GetMapping("/students/{studentId}")
    public ResponseEntity getScheduleByStudent(@PathVariable Long studentId,
@RequestParam("scheduleType") Integer scheduleType) {
        return new ResponseEntity(scheduleService.getScheduleByStudent(studentId,
scheduleType), HttpStatus.OK);
    }
}
```

ДОДАТОК Б

Відгук керівника економічного розділу

ДОДАТОК В

Перелік файлів на диску

ПЕРЕЛІК ДОКУМЕНТІВ НА МАГНІТНОМУ НОСІЇ

Ім’я файлу	Опис
Пояснювальні документи	
Кваліфікаційна робота Свириденко.docx	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
Кваліфікаційна робота Свириденко.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF
Програма	
Свириденко.rar	Архів. Містить коди програми і скомпільовану програму
Презентація	
Свириденко.ppt	Презентація кваліфікаційної роботи