Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Розробка системи для автоматизації управління Git-репозиторіями

Виконав: Стд IV курсу, групи СНс-42
спеціальності 122 Комп’ютерні науки
(шифр і назва спеціальності)

_____ Золотий Д.М.
(підпис) (прізвище та ініціали)

Керівник _____ Приймак М.В.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Шимчук Г.В.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Боднарчук І.О.
(підпис) (прізвище та ініціали)

Рецензент _____ Пастух О.А.
(підпис) (прізвище та ініціали)

Тернопіль - 2022

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«__» _____ 2022 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)

Студент Золотому Дмитру Миколайовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка системи для автоматизації управління Git-репозиторіями

Керівник роботи Приймак Микола Володимирович, д.т.н., проф. каф. КН
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «16» 03 2022 року № 4/7-161

2. Термін подання студентом завершеної роботи 12.06.2022р.

3. Вихідні дані до роботи наукові літературні джерела

4. Зміст роботи (перелік питань, які потрібно розробити)
1. Аналіз предметної області. 1.1. Використання GitLab у навчальному процесі.
1.2. Актори системи. 1.3. Функціональні вимоги до системи. 1.4. Нефункціональні вимоги.
2. Теоретична частина. 2.1. Модель предметної області. 2.2. Типи проектів. 2.3 Вибір інструментів. 2.4 Архітектура системи. 2.5 Контролери ASP.NET Core MVC. Services.
2.6 Пакет. 2.7 Взаємодія з GitLab. 2.8. Взаємодія з Git -репозиторіями
3. Практична частина. 3.1. Архівація та збирання коду проектів
3.2. Імпорт студентів. 3.3 Статистика. 3.4 Розгортання системи
4. Безпека життєдіяльності, основи хорони праці.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1. Титулка. 2. Актуальність. 3. Мета, задачі дослідження. 4. Використання GitLab у навчальному процесі. 5. Актори та функціональні вимоги до системи
6. Модель предметної області. 7. Типи проектів. 8. Вибір інструментів створення системи
9. Архітектура системи. 10. Взаємодія з GitLab та Git-репозиторіями.
10. Скрін-шоти вікон роботи системи. 12. Діаграма розгортання системи
12. Основні результати проведеного дослідження

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи хорони праці	Гурик О.Я., доцент кафедри МТ		

7. Дата видачі завдання 2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	16.03 – 19.03	*Виконано*
2.	Підбір джерел про управління Git-репозиторіями	20.03 – 15.04	*Виконано*
3.	Опрацювання джерел про системи управління		
	Git-репозиторіями	16.04 – 28.04	*Виконано*
4.	Виконання дослідження систем управління		
	Git-репозиторіями	29.04 – 10.05	*Виконано*
5	Розроблення програмного коду	11.05 – 15.05	*Виконано*
6.	Оформлення розділу «Аналіз предметної області»	15.05 – 20.05	*Виконано*
7.	Оформлення розділу «Теоретична частина»	21.05 – 26.05	*Виконано*
8.	Оформлення розділу «Практична моделювання»	27.05 – 31.05	*Виконано*
9.	Виконання завдання до підрозділу «Безпека		
	життєдіяльності, основи хорони праці»	12.05 – 18.05	*Виконано*
10.	Оформлення кваліфікаційної роботи	19.05 – 31.05	*Виконано*
11.	Нормоконтроль	01.06 – 18.06	*Виконано*
12.	Перевірка на плагіат	09.06 – 13.06	*Виконано*
13.	Попередній захист кваліфікаційної роботи	12.06 – 17.06	*Виконано*
14.	Захист кваліфікаційної роботи	22.06	

Студент _____ Золотий Д.М.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Приймак М.В.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Розробка системи для автоматизації управління Git-репозиторіями // Золотий Дмитро Миколайович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем та програмної інженерії, кафедра комп'ютерних наук, група СНс–42 // Тернопіль, 2022 // С. – 54, рис. – 24, табл. – 1, слайдів – 13, бібліогр. – 15.

Ключові слова: автоматизація процесу навчання, git, gitlab, asp.net core, веб-додаток

Кваліфікаційна робота присвячена створенню та впровадженню веб-застосунку для автоматизації керування Git -репозиторіями на основі системи GitLab з метою застосування в освітньому процесі.

У першому розділі кваліфікаційної роботи описано використання GitLab у процесі навчання. Визначено акторів та вимоги до системи.

В другому розділі роботи описано модель предметної області, визначено типи проектів в системі. Здійснено вибір програмних інструментів (ASP.NET Core, Entity Framework Core, MS SQL Server, HTML, CSS та JS -фреймворк Bootstrap, libgit2sharp, Docker). Побудована архітектура системи. Описано всі контролери та взаємодія з GitLab і Git –репозиторіями.

У третьому розділі розглянуто особливості проведення архівації та збирання коду проектів, імпорту студентів в системі, їх статистичні дані. Наведено розгортання системи з використанням платформи контейнеризації та образів Docker.

Четвертий розділ роботи присвячений важливим питанням безпеки життєдіяльності та основ охорони праці.

ANNOTATION

Development of a system for the automated management of Git-repositories // Zolotiy Dmytro// Ternopil Ivan Pul'uj National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Computer Science // Ternopil, 2022 // P. - 54, Fig. - 24, Table - 1, Slide - 13, References - 17.

Keywords: automation of the learning process, git, gitlab, asp.net core, web app

Thesis deals with the creation and implementation of a web application to automate the management of Git- repositories based on the GitLab system for use in the educational process.

The first section of the qualification describes the use of GitLab in the learning process. Actors and system requirements are defined.

The second section describes the model of the subject area, identifies the types of projects in the system. The choice of software tools (ASP.NET Core, Entity Framework Core, MS SQL Server, HTML, CSS and JS-framework Bootstrap, libgit2sharp, Docker). The architecture of the system is built. All controllers and interaction with GitLab and Git- repositories are described.

The third section discusses the features of archiving and collecting project code, importing students into the system, their statistics. The deployment of the system using the containerization platform and Docker images is given.

The fourth section of the work is devoted to important issues of life safety and basics of labor protection.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ СКОРОЧЕНЬ І ТЕРМІНІВ

API (Application Programming Interface) – прикладний програмний інтерфейс

CRUD (Create Read Update Delete) – 4 основні функції управління даними «створення, читання, оновлення і вилучення».

fork – операція системи управління репозиторіями, що створює копію репозиторію.

Git – розподілена система версій, яка дозволяє зберігати кожну нову версію проекту, надає потужні інструменти для керування.

GitLab – система управління репозиторіями коду для системи контролю версій Git.

git clone – операція git, яка дозволяє отримати локальну копію існуючого репозиторію.

git commit - операція git, яка записує зміни до репозиторію.

git push - операція git, яка відправляє у віддалений репозиторій зміни з локального.

Адм – Адміністратор.

АС – автоматизована система.

БД – база даних.

ВС – веб-сервер.

Вклд – викладач.

Кн – контейнер.

Коміт (від англ. commit – здійснювати, фіксувати) – збережений у репозиторії набір змін у програмному коді.

Пр – предмет.

ПрО – предметна область.

Репозиторій (у системах керування версій) - місце (сховище), де зберігаються всі файли разом з історією їхньої зміни та іншою службовою інформацією.

ОС – операційна система.

СКВ – система контролю версій.

Стд – студент.

ЗМІСТ

ВСТУП

Актуальність теми. На сьогоднішній день існує проблема ефективної взаємодії між Вклд та Стд, що прямо впливає на якість їхньої роботи. Для того, щоб підвищити продуктивність і загальний рівень освіти в даний час заклади освіти намагаються автоматизувати процес навчання, знизити час на виконання однотипних, рутинних завдань. Впровадження АС для підтримки навчання дозволяє скоротити час та зменшити трудовитрати для всіх учасників, залучених до навчального процесу.

Дуже часто для курсів програмування і особливо тих, де Стд доводиться працювати в команді, виникає необхідність довгострокового зберігання вихідного коду програм, організації спільної розробки програмного продукту, можливість повертатися до попередніх версій проекту. Для Вклд, на Пр якого Стд пишуть програмний код, у такому разі однією з проблем стає збір коду Стд та відстеження їх прогресу, оскільки їхні проекти можуть оформлятися по-різному, що ускладнює перевірку. Вирішити цю проблему допомагає використання СКВ - системи, яка дозволяє зберігати кілька версій одного і того ж документа, фіксувати хто і коли вносив зміни, виконувати повернення до минулих версій. Використання такої системи дозволяє відслідковувати та синхронізувати зміни у розроблюваному проекті. Даний підхід дозволяє Стд зберігати свої навчальні проекти, надає корисні скіли роботи з професійними інструментами. Для Вклд використання СКВ скорочує час на збирання та перевірку завдань від Стд, тим самим полегшуючи йому роботу.

Сьогодні існує кілька популярних СКВ, яскравим представником яких є Git [1]. Для впровадження СКВ git можна використовувати різні продукти, одним із яких є GitLab [2], котрий є одним з найпопулярніших, повністю відкритим і безкоштовним продуктом. GitLab надає різні інструменти для розробників та орієнтований здебільшого на професійну та корпоративну розробку.

Мета дослідження – забезпечити використання GitLab у процесі навчання для того, щоб дозволити Вклд більш ефективно і простіше вибудовувати

взаємодію зі Стд.

Для досягнення мети виділено ряд задач:

- проаналізувати вимоги;
- здійснити аналіз API GitLab;
- спроектувати та реалізувати систему;
- впровадити систему.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Використання GitLab у навчальному процесі

GitLab дозволяє створити для кожного Стд та Вклд індивідуальні облікові записи, за допомогою яких вони матимуть доступ до необхідних репозиторіїв. Репозиторії в системі GitLab є невід'ємною складовою проектів і не можуть існувати самостійно. Отже, управління репозиторіями відбувається через управління проектами.

Проекти в GitLab створюються в рамках простору імен, в якості якого можуть виступати користувачі або групи GitLab, і можуть мати один із наступних рівнів видимості:

- громадські (public) проекти доступні всім і вимагають аутентифікації;
- внутрішні проекти доступні всім користувачам GitLab;
- приватні (private) проекти доступні лише для обмеженого числа користувачів.

Коли проект формується в просторі імен користувача, цей користувач володітиме повним доступом до нього з можливістю надання доступу іншим користувачам. Якщо проект створюється в просторі імен групи, доступ до нього надається всім членам групи залежно від їх рівня доступу в самій групі, а також може бути явно наданий іншим користувачам.

При використанні GitLab у процесі навчання більшість користувачів не матимуть великого досвіду роботи з ним і за наявності достатньо високого доступу в системі можуть випадково чи навмисно вчинити дії, які можуть призвести до таких проблем:

- опечататися в назві репозиторію, що утруднить використання засобів автоматизації, котрі можуть використовуватися для автоматичного збору завдань та обчислення активності Стд;
- видалити репозиторій із завданням, втративши всі свої чи чужі зміни у ньому;
- видалити окремі коміти з репозиторію шляхом виконання команди git

push – force;

- надати доступ до свого репозиторію третім особам;
- не надати Вклд доступ до потрібного репозиторію або позбавити його цього доступу.

У зв'язку з цим пропонується максимально обмежити Стд права в GitLab, залишивши їм мінімальний набір прав, достатній для роботи з потрібними їм репозиторіями. Для цього Стд відключається можливість створювати проекти та групи та створюється спеціальна особиста група для його проектів. Використання особистої групи дозволяє виключити необхідність створювати проекти у просторі імен користувача та дозволяє обмежити йому права в його проектах (інакше, у Стд був би повний доступ до його особистих проектів).

Для організації командної роботи Стд над проектом Вклд розбиває їх на команди. Склади команд визначаються індивідуально кожному за проекту. Створюється ще одна група GitLab - командна, призначена для зберігання всіх репозиторіїв різних команд, що виконують завдання по Пр. Кожна команда працює над своїм завданням ізольовано від інших команд, де кожен учасник може вносити зміни до репозиторію своєї команди.

Використання цього підходу дозволяє в потрібному обсязі обмежити Стд права в GitLab і зводить до мінімуму можливість Стд зламати що-небудь в GitLab. За винятком втрати доступу до облікового запису (у випадку, якщо Стд забув свій пароль і не налаштував або втратив доступ до електронної пошти для скидання пароля) Стд може вчинити такі деструктивні дії:

- вихід із особистої групи;
- вихід із групи, створеної для досліджуваного ним Пр;
- вихід із командного проекту.

Дані дії можуть бути легко блокуватися Адм і не призводять до втрати даних.

Аналогічно доступ може бути обмежений і Вклд. Йому також відключається можливість створювати проекти самостійно і натомість створюється по дві (три, якщо необхідна командна робота Стд) групи для кожного Пр, для яких використовуватиметься GitLab. Одна з цих груп

використовуватиметься лише Вклд даного Пр для розміщення заготовок початкового коду, авторських рішень, чернеток матеріалів та інших даних, які не повинні бути доступні Стд. Надалі ця група згадуватиметься як приватна група Пр. Інша група буде призначена для публікації інформації, яка має бути доступна Стд. Крім Вклд до цієї групи матимуть доступ Стд, які вивчають цей Пр. Вподальшому група буде йменуватися як публічна група Пр. У Пр, крім публічної та приватної GitLab групи, може додаватися ще одна група - командна, якщо на курсі запланована командна робота. До цієї групи будуть додані Вклд Пр, а Стд надаватиметься доступ лише до конкретних проектів (інакше Стд, доданий до групи, може бачити всі проекти групи). Такий підхід було прийнято з метою обмеження можливості членів однієї команди бачити роботу інших команд. При цьому, технічно, всі три групи матимуть приватний рівень видимості і тільки їхні члени їх бачитимуть і матимуть до них доступ.

В результаті виходить наступний набір обмежень:

– ані Стд, ані Вклд не можуть самостійно створювати нові проекти, а також керувати існуючими;

– для кожного Стд створюється його особиста група, до якої матиме доступ лише він. Ця група призначена для проектів, над якими потрібно працювати Стд;

– для кожного Пр створюються приватна та публічна група Пр. Членами приватної групи будуть лише Вклд даного Пр, членами публічної групи - Вклд та Стд, які вивчають цей Пр;

– для можливості перевірки Вклд надається явний доступ до тих проектів, які відносяться до їх Пр;

– для командних робіт Пр опціонально створюється командна група, членами якої будуть лише Вклд даного Пр. Стд буде надано доступ до конкретних проектів, що створюються всередині такої групи.

При використанні даного підходу виникає наступна проблема - створення репозиторіїв доступне тільки Адм GitLab і, в результаті, крім завдань з адміністрування GitLab (управління користувачами та груп), Адм додаються завдання з супроводу ПР. За запитом Вклд Адм повинен буде створювати

потрібні репозиторії, поширювати їх Стд та виконувати інші дії, які має виконувати Вклд.

Одим із варіантів розв’язання цієї проблеми – надання Вклд прав Адм GitLab. Це дозволить йому самостійно виконувати всі необхідні операції, але ускладнить використання одного екземпляра GitLab для кількох Пр. Використання декількох екземплярів GitLab (по одному для кожного Пр), у свою чергу, завдасть незручностей Стд (у Стд будуть різні облікові записи в різних примірниках GitLab) і збільшить технічні вимоги до розміщення цих екземплярів. Крім цього, супровід екземпляра GitLab з урахуванням перерахованих вище обмежень може бути досить трудомістким завданням.

Іншим способом вирішення власне такої проблеми є розробка саме такого інструменту, котрий би дозволив делегувати необхідні права Вклд, не дозволяючи їм їх перевищувати. В результаті чого Вклд отримали можливість створювати репозиторії, поширювати їх Стд, збирати готові рішення в рамках тих Пр, які вони викладають (не маючи доступу до інших Пр і Стд, які не освоюють їх Пр).

Основними завданнями даної є проектування і реалізація такої системи.

1.2 Актори системи

При роботі з системою GitLab можна виділити 3 види акторів: Стд, Вклд та Адм.

Стд взаємодіє з GitLab наступним чином. Він отримує завдання від Вклд, виконує його та відправляє виконане завдання. Стд може переглядати матеріали курсу, які публікує Вклд. Стд може працювати над командними проектами разом з іншими Стд.

Вклд готує завдання (домашні та контрольні) для Стд і публікує їх. Після того як Стд надіслали виконані завдання, Вклд може переглянути їх роботи та поставити оцінки. Вклд може розбивати Стд на команди. Вклд також може розміщувати матеріали по Пр, доступні для читання всім Стд і додавати в приватну групу своє рішення завдань.

Адм, при початковому налаштуванні системи, додає необхідні факультети. Щороку один раз він додає нові групи та заносить до них нових Стд. Потім додає перелік Пр. Адм також додає Вклд, яких він пов'язує із потрібними Пр. Адм може керувати всіма Стд, Вклд, факультетами, Пр, проектами та командами.

Серед вищезазначених акторів Стд достатньо вбудованих можливостей GitLab, тому система, що розробляється, буде орієнтована тільки на Вклд та Адм.

1.3 Функціональні вимоги до системи

Були сформовані такі функціональні вимоги, визначені як варіанти використання

Стд може:

- переглядати матеріали на Пр;
- виконувати домашні та контрольні роботи;
- виконувати командні завдання;
- додавати свої матеріали та нотатки.

Вклд може:

- керувати проектами на Пр (CRUD);
- записувати Стд на Пр;
- керувати Стд, записаними на його Пр;
- керувати Вклд та їх правами у межах Пр та проекту;
- керувати командами проекту;
- керувати репозиторіями проекту;
- дивитися статистику роботи Стд та команд;
- архівувати проекти, Стд та команди;
- збирати код проектів;
- публікувати матеріали по Пр;
- викладати домашні та контрольні роботи Стд та командам;
- перевіряти домашні та контрольні роботи Стд та команд.

Адм може:

- управляти проектами (CRUD);
- керувати Пр (CRUD);
- керувати Вклд (CRUD);
- керувати Стд (CRUD), зокрема імпортувати Стд;
- керувати групами (CRUD);
- управляти факультетами (CRUD);
- керувати синхронізацією з GitLab;
- керувати користувачами.

Варіанти використання для Стд три останні для Вклд реалізуються самою системою GitLab. Інші варіанти використання будуть реалізовані в системі, що розробляється.

Розглянемо докладніше сценарії найважливіших варіантів використання.

1. Імпортувати Стд. Основний актор: Адм**.**

Основний успішний сценарій:

- Адм тисне кнопку «Import» на Стд сторінці;
- система здійснює відкривання сторінки імпорту Стд, на якій відображає поля для вибору факультету та групи, багаторядкове поле для занесення списку Стд;
- Адм виконує вибір факультету;
- система відображає групи обраного факультету;
- Адм вибирає групу, заносить список Стд та натискає кнопку «Next»;
- система розбирає внесені дані та відображає результат розбору у вигляді списку Стд, які будуть імпортовані, а також пропонує для кожного Стд створити користувача GitLab та особисту групу;
- Адм підтверджує, що Стд, які імпортуються, потрібно створити користувача та особисту групу GitLab;
- система зберігає Стд та створює їм користувача та особисту групу GitLab.

Розширення:

- у Стд, що імпортується, вже існує користувач і особиста група GitLab: Адм скасовує імпортованим Стд створення юзера та особистої групи GitLab;

система зберігає Стд.

2. Створити Стд. Основний актор: Адм.

Основний успішний сценарій:

– Адм тисне «Create»- кнопку на сторінці Стд;

– система здійснює відкриття сторінки створення Стд, на якій відображає поля для занесення загальної інформації про Стд, поля для вибору факультету та групи;

– Адм заносить загальну інформацію, вибирає факультет;

– система відображає групи обраного факультету;

– Адм обирає групу;

– система пропонує для кожного Стд створити користувача GitLab та особисту групу;

– Адм підтверджує, що Стд потрібно створити користувача і особисту групу GitLab;

– система відображає попередньо заповнені поля для створення користувача та групи GitLab;

– Адм проводить зміни (за необхідності), і натискає кнопку Create;

– система зберігає Стд та створює їм користувача та особисту групу GitLab.

Розширення:

– у Стд вже існує користувач і особиста група GitLab: Адм скасовує створюваному Стд створення юзера та особистої групи GitLab; система зберігає Стд.

3. Створити Пр. Основний актор: Адм

Основний успішний сценарій:

– Адм здійснює натискання кнопки «Create» на вкладці Subjects на сторінці потрібного факультету;

– система відкриває власне сторінку для створення нового Пр, де можна задати назву Пр та вибрати Вклд;

– Адм задає ім'я та вибирає Вклд, натискає кнопку «Create»;

– система зберігає новий Пр та відображає інформацію про Пр;

– Адм переходить на вкладку Settings на сторінці створеного Пр і натискає кнопку Create GitLab groups;

– система створює дві групи для Пр – приватну та публічну.

Розширення:

– На факультет не додано жодного Вклд:

– система відображає "No teacher available";

– Адм визначає лише назву Пр, тисне на кнопку «Create»;

– система зберігає новий Пр та відображає інформацію про нього;

– Адм йде на сторінку Teachers, потім тисне по кнопці Create;

– система виконує відкриття сторінки для створення Вклд.

– Адм вказує ім'я, вибирає потрібний факультет та натискає «Create»;

– система зберігає «свіжого» Вклд та відображає інформацію про нього;

– Адм йде на сторінку потрібного Пр, на вкладку Teachers і клікає по кнопці «Add teacher»;

– система проводить відкриття сторінки для додавання Вклд на Пр, де відображається список Вклд факультету;

– Адм вибирає потрібних Вклд та натискає «Add»;

– система додає обраних Вклд на Пр. Додає відповідних вибраним Вклд користувачів GitLab у публічну та приватну групу GitLab з цього Пр.

4. Записати Стд на Пр. Основний актор: Вклд.

Основний успішний сценарій:

– Вклд натискає кнопку «Add student» на вкладці «Students» на сторінці потрібного Пр;

– система відкриває сторінку для додавання Стд на Пр, де відображається список груп, всередині кожної з них розкривається підсписок Стд цієї групи;

– Вклд вибирає або всю групу, заносячи при цьому всіх Стд цієї групи або окремих Стд і тисне «Add»;

– система додає обраних Стд на Пр. Додає відповідних вибраним Стд користувачів GitLab у публічну групу GitLab з цього Пр.

На додачу до вже наведених сценаріїв у системі, котра розробляється, будуть ще такі сценарії: Створити проекти видів «Домашня робота», «Контрольна робота», «Матеріали Викладача», «Матеріали Студента», «Командна домашня робота», «Командна контрольна робота», «Матеріали команди».

1.4 Нефункціональні вимоги

Для системи, що розробляється, були визначені такі нефункціональні вимоги:

– підтримка ОС сімейства UNIX, оскільки GitLab підтримує ці ОС;
– підтримка GitLab версії 11.

РОЗДІЛ 2. ТЕОРЕТИЧНА ЧАСТИНА

2.1 Модель предметної області

На рис. 2.1 наведено повна модель ПрО.

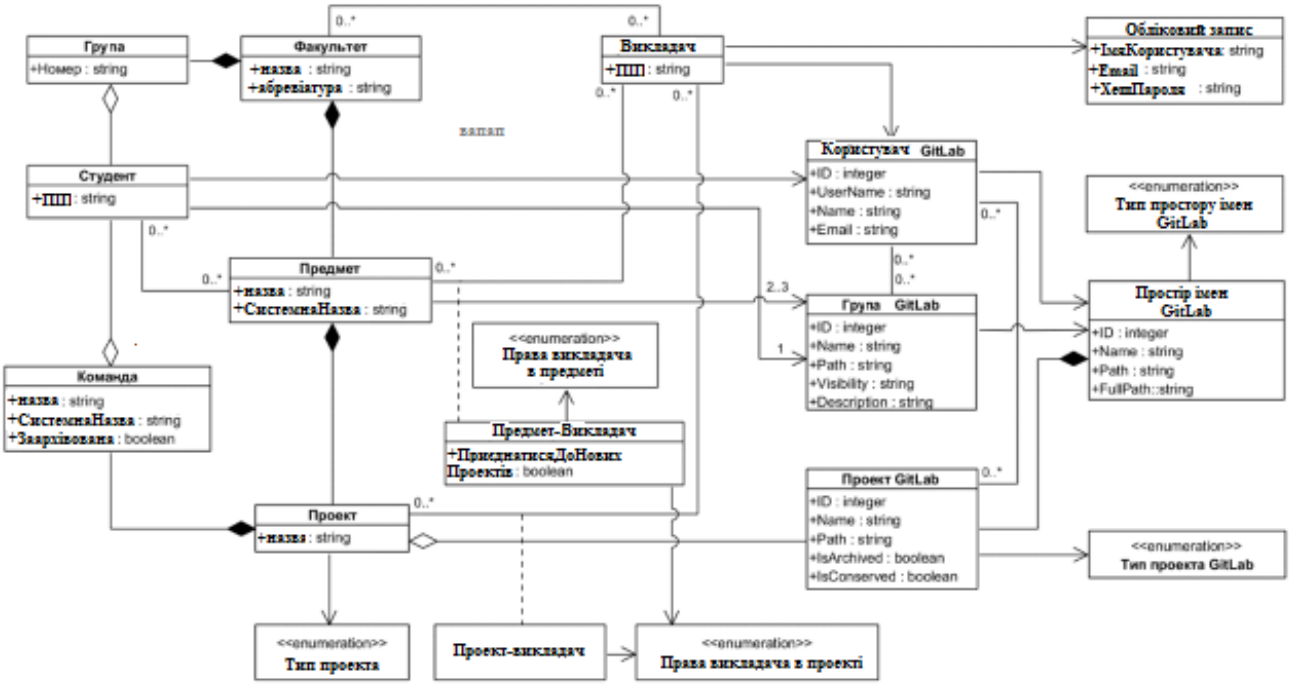

Рисунок 2.1 – Модель ПрО

Опишемо докладніше кожен із класів моделі з рис. 2.1.

Факультет – зберігання інформації про факультети навчального закладу.

Група – збереження інформації про навчальні групи, на які діляться Стд.

Викладач – зберігає інформацію про Вклд.

Студент – збереження інформації про Стд.

Предмет – зберігається інформація про Пр.

Проект – збереження інформації про проекти. Під проектом у системі, що розробляється, розуміється якась збірна сутність над домашніми роботами, контрольними роботами, матеріалами Вклд та Стд.

Тип проекту – перелік можливих типів проекту: домашня робота, контрольна робота, матеріали Вклд, матеріали Стд, командна домашня робота,

командна контрольна робота, матеріали команди.

Команда - зберігання інформації про команди проекту та Стд, що входять до цих команд.

Предмет-викладач - присутність Вклд в Пр, та формує права Вклд в межах предмета.

Права викладача у предметі - набір прав, котрими може володіти Вклд у межах Пр: немає прав, створювати проекти, управляти Вклд, управляти Стд, управляти будь-якими проектами Пр, повний доступ.

Проект-викладач - присутність Вклд у проекті, а також задає права Вклд в рамках проекту.

Права викладача у проекті - перерахування можливих прав, які може мати Вклд у рамках проекту: немає прав, керувати репозиторіями, переглядати репозиторії Стд, керувати проектом, повний доступ.

Користувач GitLab - зберігання інформації про фактичних користувачів GitLab, включає їх ідентифікатори та дані, потрібні системі для співпраці з власне GitLab.

Група GitLab - збереження інформації про фактичні групи GitLab.

Простір імен GitLab - зберігання інформації про фактичні простори імен GitLab.

Тип простору імен GitLab - перелік, що описує можливі типи просторів імен в GitLab.

Проект GitLab - зберігання інформації про фактичні проекти GitLab, що містить їх ідентифікатори та іншу інформацію, необхідну розроблюваній системі для взаємодії з GitLab.

Тип проекту GitLab - перерахування, що визначає призначення проекту GitLab як репозиторія, в рамках проекту системи, що розробляється.

2.2 Типи проектів

Розглянемо докладніше типи проектів, що визначаються системою, що розробляється .

Домашня робота – надає Вклд можливість підготувати завдання для Стд, підготувати початковий код для них, розіслати завдання та код Стд шляхом копіювання репозиторію в їхні особисті групи і потім отримати виконані власне ними завдання для перевірки (рис. 2.2).

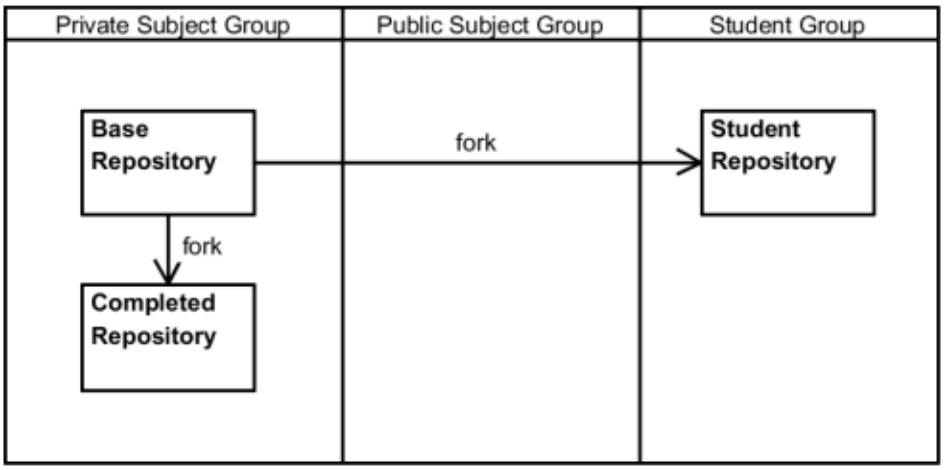

Рисунок 2.2 – Проект типу «Домашня робота»

Контрольна робота - дозволяє Вклд запропонувати різноманітні варіанти завдань для Стд, які будуть розіслані Стд індивідуально після початку контрольноого заходу (рис. 2.3).

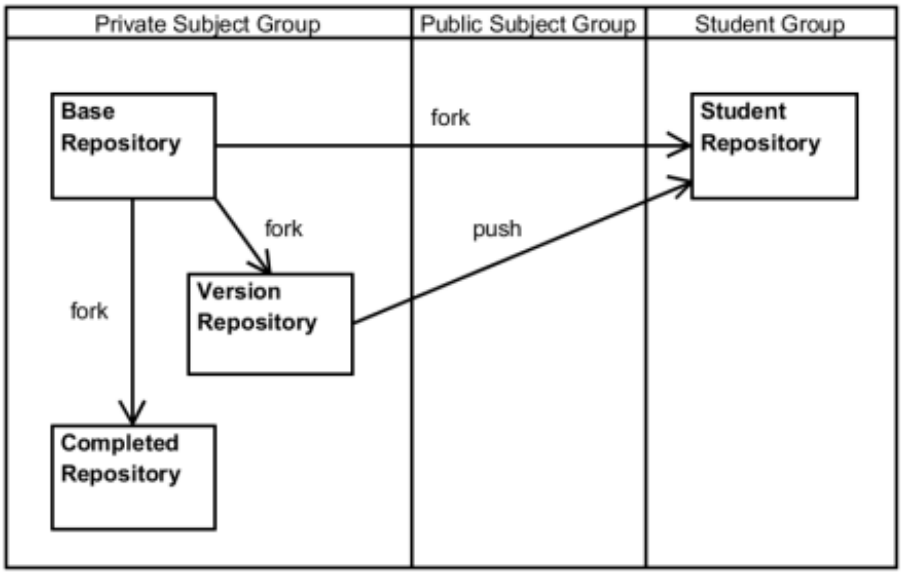

Рисунок 2.3 – Проект типу «Контрольна робота»

Матеріали викладача – забезпечує для Вклд можливість публікування матеріалів Стд (рис. 2.4).

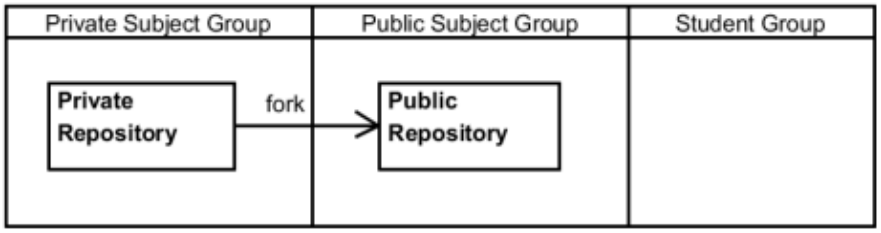

Рисунок 2.4 – Проект типу «Матеріали викладача»

Матеріали студента - тип проекту, що дозволяє Стд зберігати свої нотатки в особистих репозиторіях (рис . 2.5).

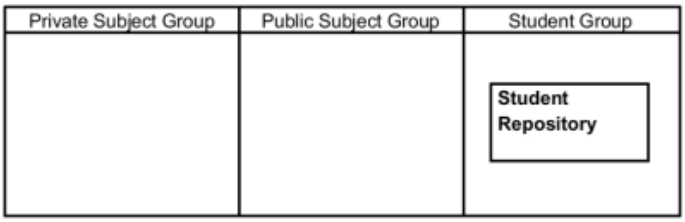

Рисунок 2.5 – Проект типу «Матеріали студента»

Командна домашня робота - тип проекту, який дозволяє Вклд підготувати завдання для команд, підготувати початковий код для них, розбити Стд на команди, розіслати завдання та код командам шляхом копіювання репозиторію в командну групу і потім зібрати виконані Стд завдання для перевірки (рис. 2.6).

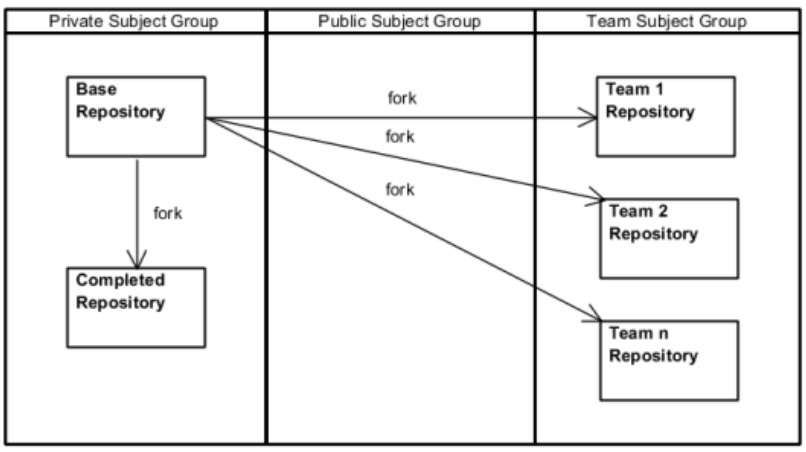

Рисунок 2.6 – Проект типу «Командна домашня робота»

Командна контрольна робота - тип проекту, що дозволяє Вклд підготувати різні варіанти завдань для команд, які будуть розіслані кожній команді після початку контрольної (рис. 2.8).

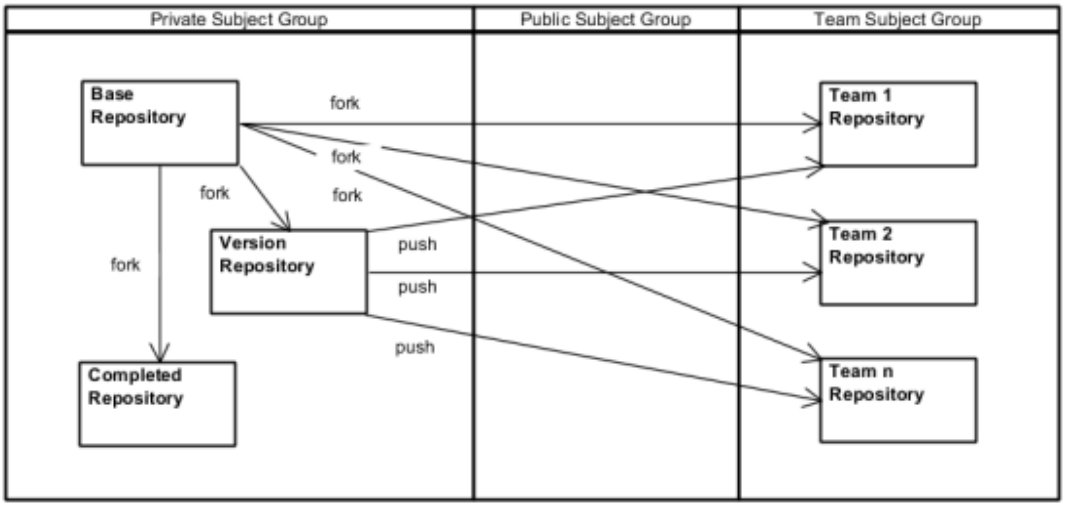

Рисунок 2.8 – Проект типу «Командна контрольна робота»

Матеріали команди - тип проекту, що дозволяє командам зберігати свої нотатки у командних репозиторіях (рис. 2.9).

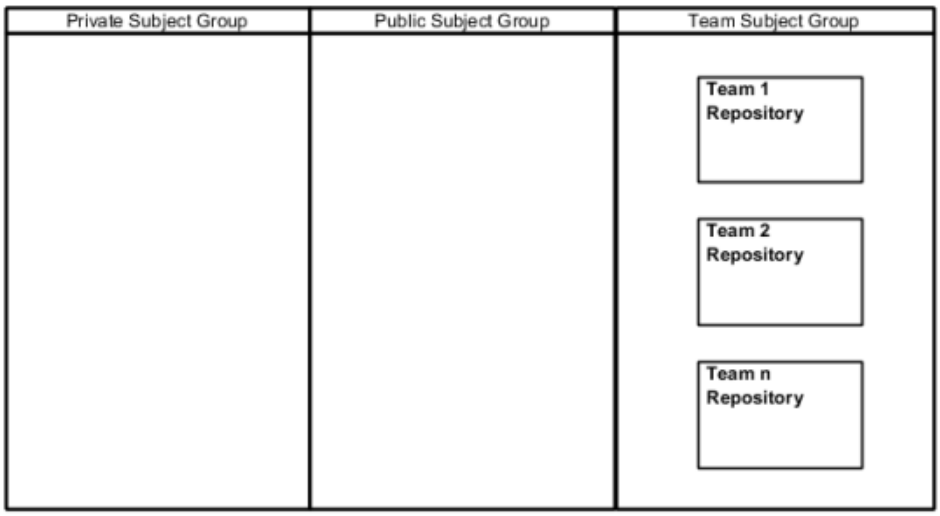

Рисунок 2.9 – Проект типу «Матеріали команди»

2.3 Вибір інструментів

Як основний фреймворк був обраний ASP.NET Core - кросплатформовий,

високопродуктивний, з відкритим кодом [3]. Він дає змогу кодити веб-програми із застосуванням патерна Model View-Controller (MVC) [4] мовою програмування C#.

Для роботи з даними був обраний Entity Framework Core [5] - кросплатформний об'єктно-реляційний перетворювач, що дозволяє визначати модель даних набором класів, на підставі яких буде створена БД та надає об'єктний інтерфейс для доступу до даних, що зберігаються в отриманій БД.

Для безпосереднього зберігання даних була обрана СУБД Microsoft SQL Server, яка також є кросплатформною і повністю підтримується Entity Framework Core.

Такий вибір інструментів дозволяє розробити систему у вигляді кросплатформенного веб-додатка, котрий потім може бути розгорнуто як на серверах під управлінням ОС Windows чи Linux.

Крім цього, були використані:

– HTML, CSS та JS -фреймворк Bootstrap 3.3 [6];

– бібліотека libgit2sharp [7], що надає інструменти для роботи з git-репозиторіями.

Для розгортання системи буде застосовано Docker- контейнерування.

2.4 Архітектура системи

З урахуванням вибраного фреймворку, розроблювану систему можна розбити на наступні пакети, представлені на діаграмі на рис. 2.10.

Розглянемо докладніше, що містить кожен із них:

– Controllers - усі реалізовані в рамах розроблюваної системи контролери ASP.NET Core MVC;

– Models - класи, що представляють модель даних і модель представлення системи, що розробляється;

– Views - представлення ASP.NET Core MVC, виконані у вигляді шаблонів Razor, з використанням патерна «Подання шаблону» [4];

– Services - допоміжні служби, що використовуються контролерами;

– GitLabApi - реалізацію клієнта API GitLab, а також необхідні йому моделі.

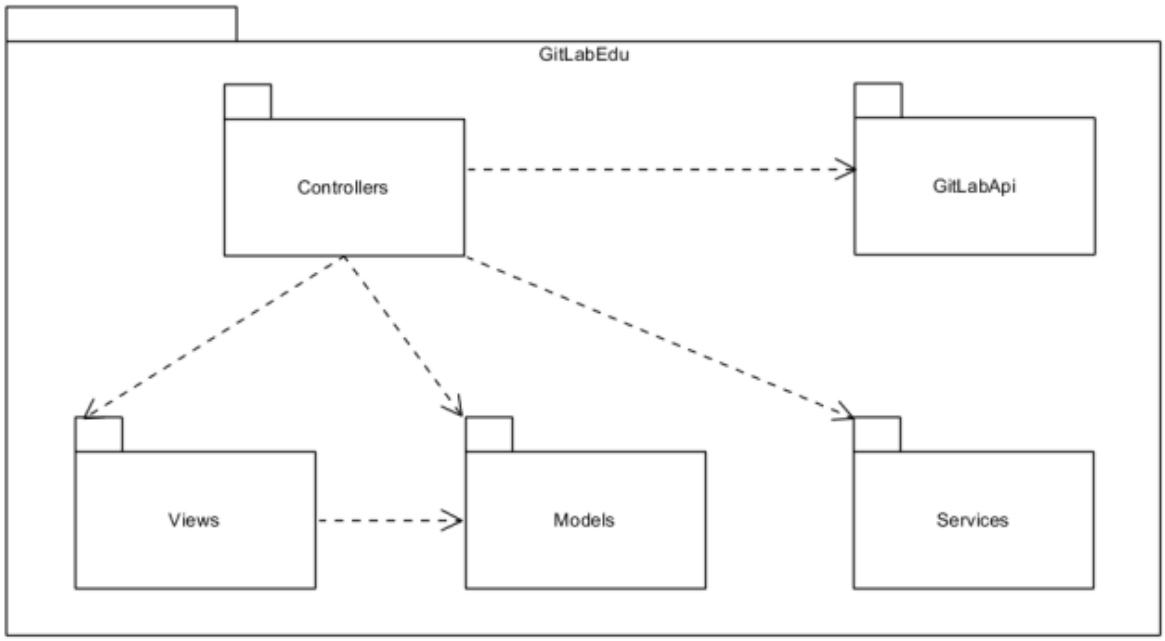

Рисунок 2.10 – Пакети системи, котра розробляється

На діаграмі, зображеній на рис. 2.11, показаний приклад реалізації патерну MVC з прикладу сутності «Предмет». Клас SubjectsController успадковується від абстрактного класу Controller, що надається фреймворком ASP.NET Core MVC . У даному класі реалізуються методи-обробники різних запитів, які можуть, наприклад, звертатися до БД за допомогою EntityFramework Core. Для цього вони використовують об'єкт класу ApplicationDbContext (успадкований від абстрактного класу DbContext, що надається EntityFramework Core), за допомогою якого можна робити CRUD -операції над сутностями ПрО (клас Subject). Результатом опрацювання запиту є представлення ASP.NET Core MVC, виконане у вигляді шаблону Razor (клас Views_Subjects_Details). Для забезпечення доступу представлення до даних, йому може бути передана модель, якою можуть виступати як класи моделі ПрО (клас Subject), так і спеціальні класи моделі представлень (клас SubjectDetails-Model).

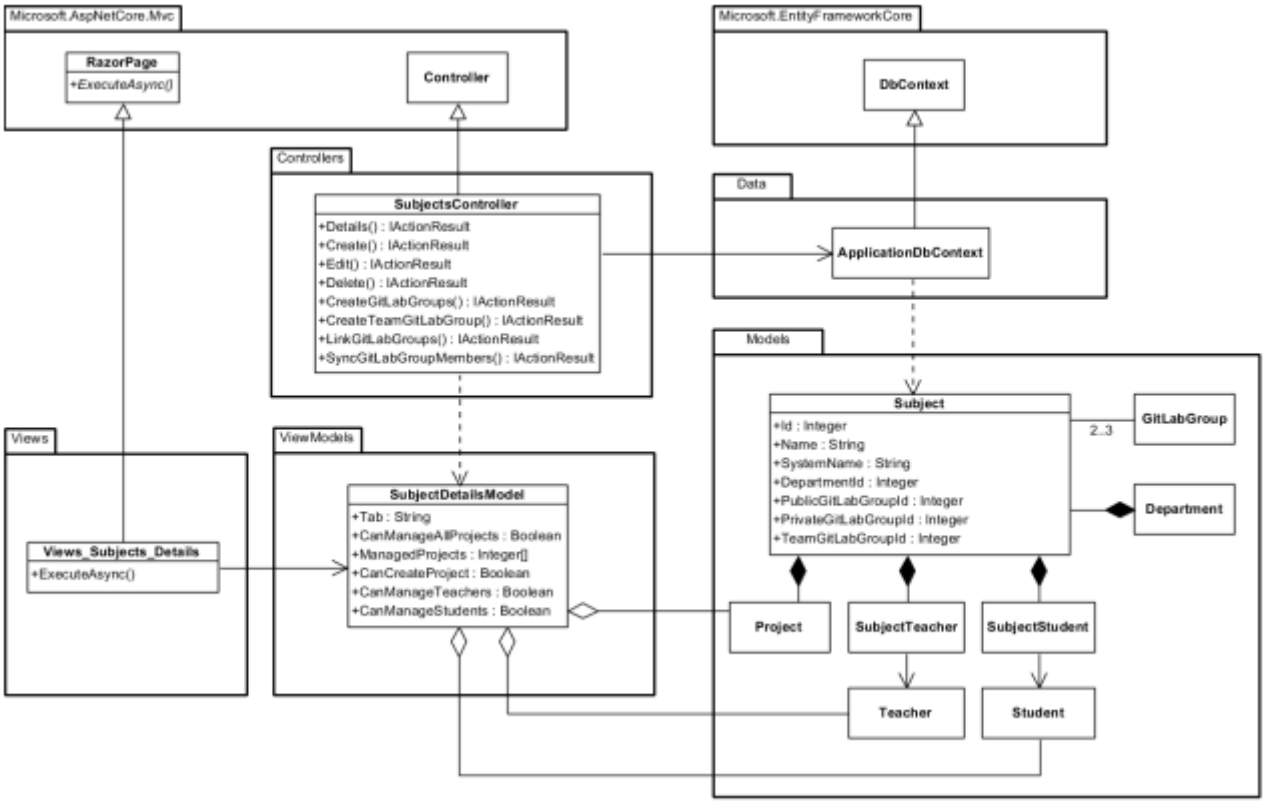

Рисунок 2.11 – Реалізація патерну MVC на прикладі сутності «Предмет» в ASP.NET Core MVC

2.5 Контролери ASP.NET Core MVC

Основні контролери для ПрО:

- DepartmentsController - дозволяє Адм управляти факультетами;
- GroupsController – дає змогу Адм управляти групами;
- TeachersController - дозволяє Адм керувати Вклд;
- DepartmentTeachersController - забезпечує зв'язок між Вклд та факультетом, що дозволяє Адм додавати та видаляти Вклд на / з факультет;
- SubjectsController – дає змогу Адм управляти Пр.
- TeacherSubjectsController - забезпечує зв'язок між Вклд та Пр, що дозволяє Адм додавати Вклд на Пр.
- StudentSubjectsController - забезпечує зв'язок між Стд та Пр, що дозволяє Вклд заносити Стд на свій Пр;
- ProjectsController - дозволяє Вклд керувати проектами;
- ProjectTeachersController - забезпечує зв'язок між проектом і Вклд, що

дозволяє Адм або Вклд з необхідними правами додавати Вклд в конкретний проект;

– ProjectTeamsController- дозволяє Вклд керувати командами;

– UsersController- дає змогу Адм керувати користувачами системи, що розробляється;

Контролери для GitLab:

– GitLabController - забезпечує навігацію за користувачами, групами та просторами імен у GitLab;

– GitLabUsersController - дозволяє переглядати користувачів GitLab, а також проводити їхню синхронізацію;

– GitLabGroupsController – дає змогу переглядати групи GitLab та здійснювати їх синхронізацію;

– GitLabNamespacesController - дозволяє переглядати для простори імен GitLab, а також проводити їх синхронізацію;

– GitLabProjectsController – дає змогу переглядати проекти GitLab, а також проводити їхню синхронізаці.

Інші контролери:

– AccountController – відповідає за всю логіку аутентифікації користувачів;

– ManageController - дозволяє користувачеві управляти своїм обліковим записом (наприклад, змінювати адресу e-mail, пароль …);

– SentEmailsController – дає змогу Адм переглянути всі електронні листи, які надсилалися користувачам;

– SubjectReportTemplatesController - дозволяє Вклд керувати шаблонами звітів на Пр.

2.6 Пакет Services

На рис. 2.12 зображено пакет Services, а також різні класи системи, які використовують служби цього пакета.

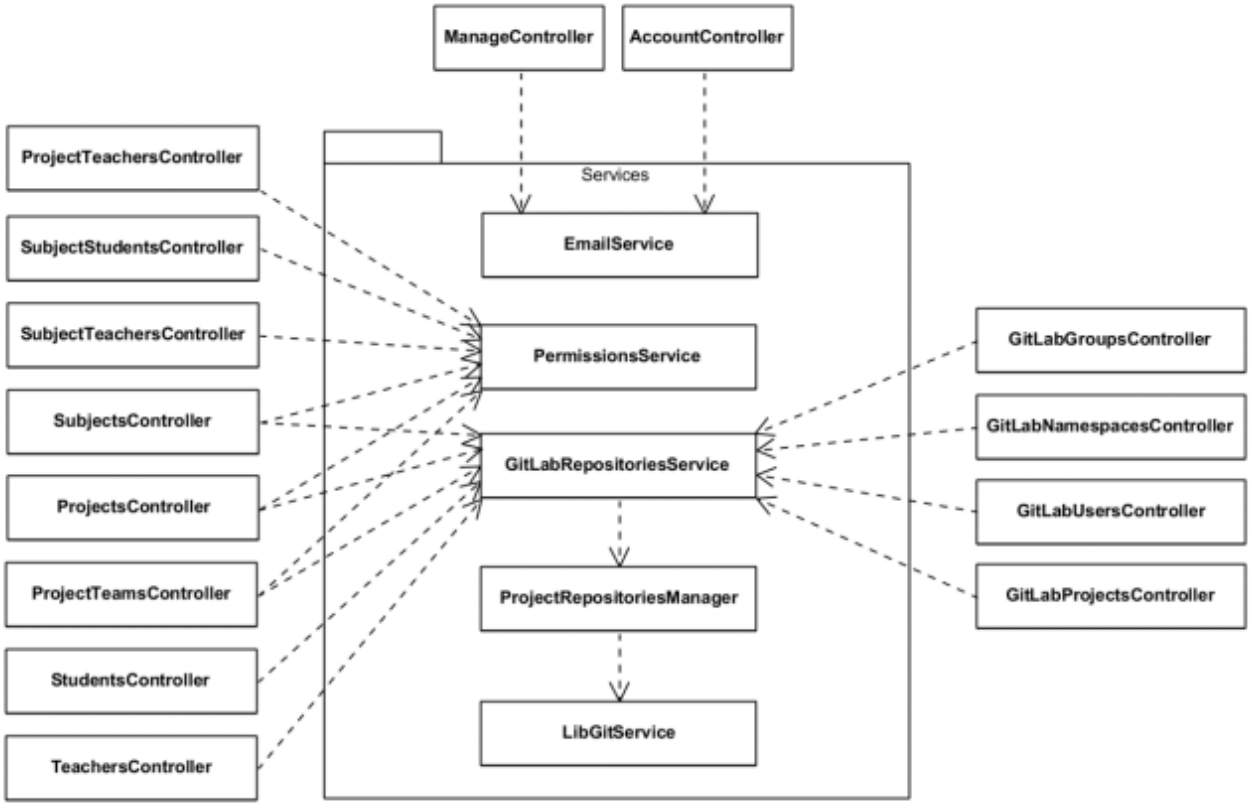

Рисунок 2.12 – Пакет Services

Пакет Services містить такі допоміжні служби.

Email Service - служба по роботі з електронною поштою, необхідна для підтвердження облікового запису, відновлення та зміни паролю. Для реалізації такої служби використали хмарний сервіс Amazon SES (Simple Email Service). Він дає API для відправлення та одержання електронних повідомлень і не вимагає розгортання власної поштової інфраструктури. Для інтеграції з цим сервісом був використаний офіційний NuGet - пакет AWSSDK.SimpleEmail від компанії Amazon, який надає клас AmazonSimpleEmailServiceClient, що надає доступ до необхідних функцій Amazon SES. Для коректної роботи цієї служби необхідний акаунт Amazon Web Services з повноцінним доступом до служби Amazon SES, а також підтверджену e-mail адресу або підтверджене доменне ім'я, яке буде використано для надсилання листів. У налаштуваннях системи необхідно вказати Access Key та Secret Key облікового запису AWS, а також адресу відправника (дана адреса має бути або підтверджена явно, або належати до підтвердженого доменного імені).

PermissionsService - служба, яка перевіряє права користувача у системі. У системі, що розробляється, необхідно зробити гнучке настроювання прав Вклд в рамках Пр цілком, а також у рамках конкретних проектів. Установка прав дозволяє обмежувати можливості Вклд у Пр (керувати Вклд, Стд, проектами) та у проекті (управління проектом, репозиторіями), а також повністю приховати окремі проекти від Вклд.

В рамках Пр Вклд може мати наступний набір прав:

– створювати проекти - дозволено створювати проекти в рамках Пр;

– керувати Вклд - дозволено додавати та видаляти Вклд на Пр, а також задавати їм права в рамках Пр;

– керувати Стд - дозволено додавати та видаляти Стд на Пр;

– керувати будь-якими проектами Пр - дозволено керувати проектами Пр, включаючи ті, що створювалися іншим Вклд;

– повний доступ - має всі перераховані вище права. Також Вклд можна додати можливість автоматичного додавання до нових проектів, які створюватимуться в рамках Пр. При цьому можна налаштувати йому стандартні права в нових проектах.

У рамках проекту Вкалд може мати наступний набір прав:

– керувати репозиторіями - дозволено в рамках проекту створювати GitLab репозиторії;

– переглядати репозиторії Стд - надано доступ до особистих репозиторій Стд, створених у рамках проекту;

– керувати проектом - дозволено змінювати налаштування проекту, керувати командами та іншими Вклд проекту;

– повний доступ - має всі перераховані вище права.

Служба PermissionsService реалізує методи перевірки прав Вклд у Пр та у проекті: HasSubjectPermissions, HasProjectPermissions.

– GitLabSyncService - служба, що відповідає за синхронізацію даних із GitLab. Ця служба дозволяє синхронізувати основні сутності GitLab: користувачів, груп, проектів та просторів імен. Методи цієї служби призначені для оновлення ЮД на основі змін, які були проведені в GitLab.

– ProjectRepositoriesManager- служба, що дозволяє керувати репозиторіями в GitLab.

У контролері ProjectsController відбувається активна робота з репозиторіями GitLab: створення репозиторіїв різних типів, їх архівація та блокування, збір коду проекту, копіювання та відправлення проекту Стд та командам. Щоб не робити контролер громіздким, вся логіка управління репозиторіями та їх взаємодії з GitLab була винесена до служби ProjectRepositoriesManager.

– LibGitService - служба, що представляє фасад [8] до бібліотеки libgit2sharp, що дозволяє виконувати операції клонування та надсилання змін.

2.7 Взаємодія з GitLab

Крім веб-інтерфейсу, через який користувачі зазвичай взаємодіють з GitLab, він також надає REST API, призначене для управління GitLab з різних засобів автоматизації [9]. Дане API і буде використовуватися системою, що розробляється, для виконання різних дій з проектами, користувачами та групами, які вимагають прав Адм. Для цього, системі, котра розробляється, буде надано private access token Адм, який дозволить їй виконувати дії від його імені.

У системі був реалізований клієнт API GitLab, котрий надає можливість викликати необхідні методи для управління користувачами, групами, просторами імен та проектами GitLab.

Доступ до GitLab буде здійснюватися великою кількістю об'єктів системи: різними контролерами і службами. Щоб звертатися до зовнішньої системи було зручніше, було застосовано архітектурний патерн «Шлюз» [4]. Клас GitLabApi-Client виступає в ролі шлюзу, який містить у собі всю логіку роботи з GitLab, а також надає зручний інтерфейс для внутрішніх об'єктів програми, які будуть з ним взаємодіяти. Такий підхід дозволить ізолювати програму від зовнішньої служби GitLab, що підвищує рівень повторного застосування коду, полегшує внесення майбутніх змін. На рис. 2.13 показана діаграма пакетів, що показує структуру і взаємозв'язок пакета GitLabApi з іншими пакетами.

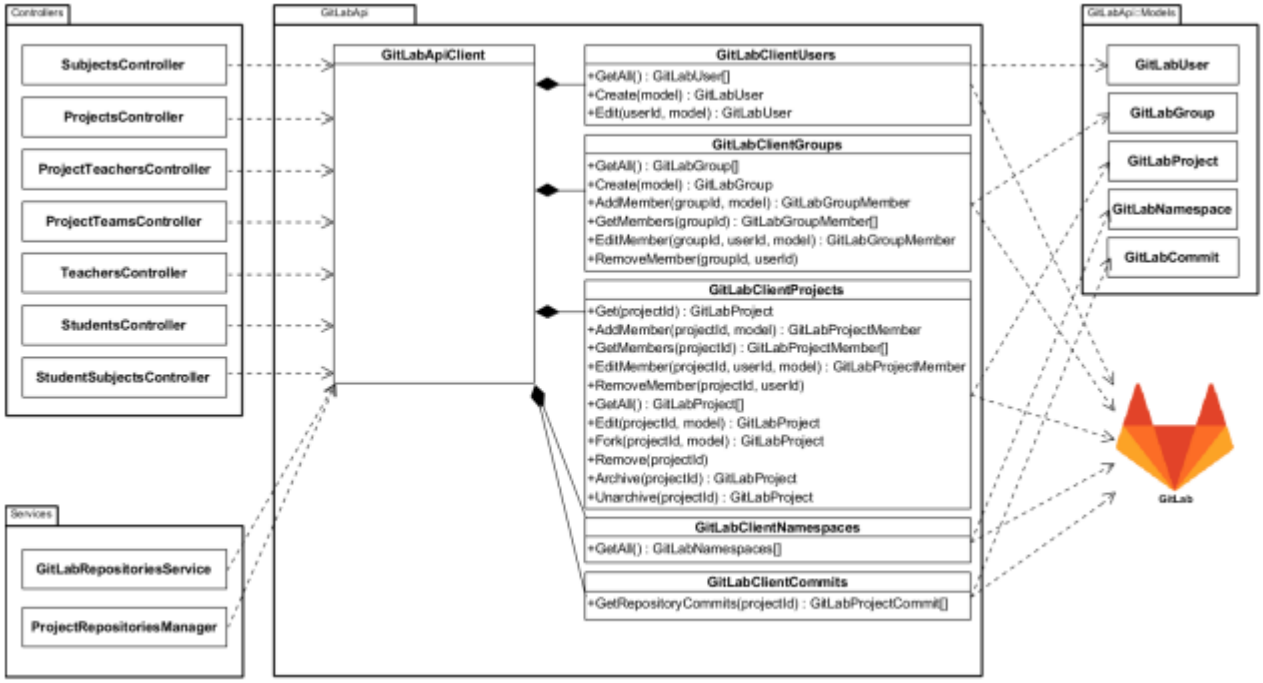

Рисунок 2.12 – Діаграма пакетів GitLabAPI

Пакет GitLabApi містить такі класи, кожен з яких відповідає за окрему підмножину методів GitLab.

– GitLabApiClientUsers - керування користувачами GitLab;

– GitLabApiClientGroups - управління групами GitLab та членами цих груп;

– GitLabApiClientProjects - управління проектами GitLab та членами цих проектів, копіювати (fork), архівувати та розархівувати проекти;

– GitLabApiClientNamespaces - управління просторами імен GitLab;

– GitLabApiClientCommits - керування коммітами.

На діаграмі послідовності (рис. 2.13) зображено процес створення базового репозиторію проекту, призначеного для розміщення початкового коду. Даний репозиторій створюється в приватній групі Пр і доступний тільки Вклд, на основі даного репозиторію створюватимуться подальші репозиторії проекту. Даний тип репозиторію недоступний у проектах типу «Матеріали студента» та «Матеріали команди».

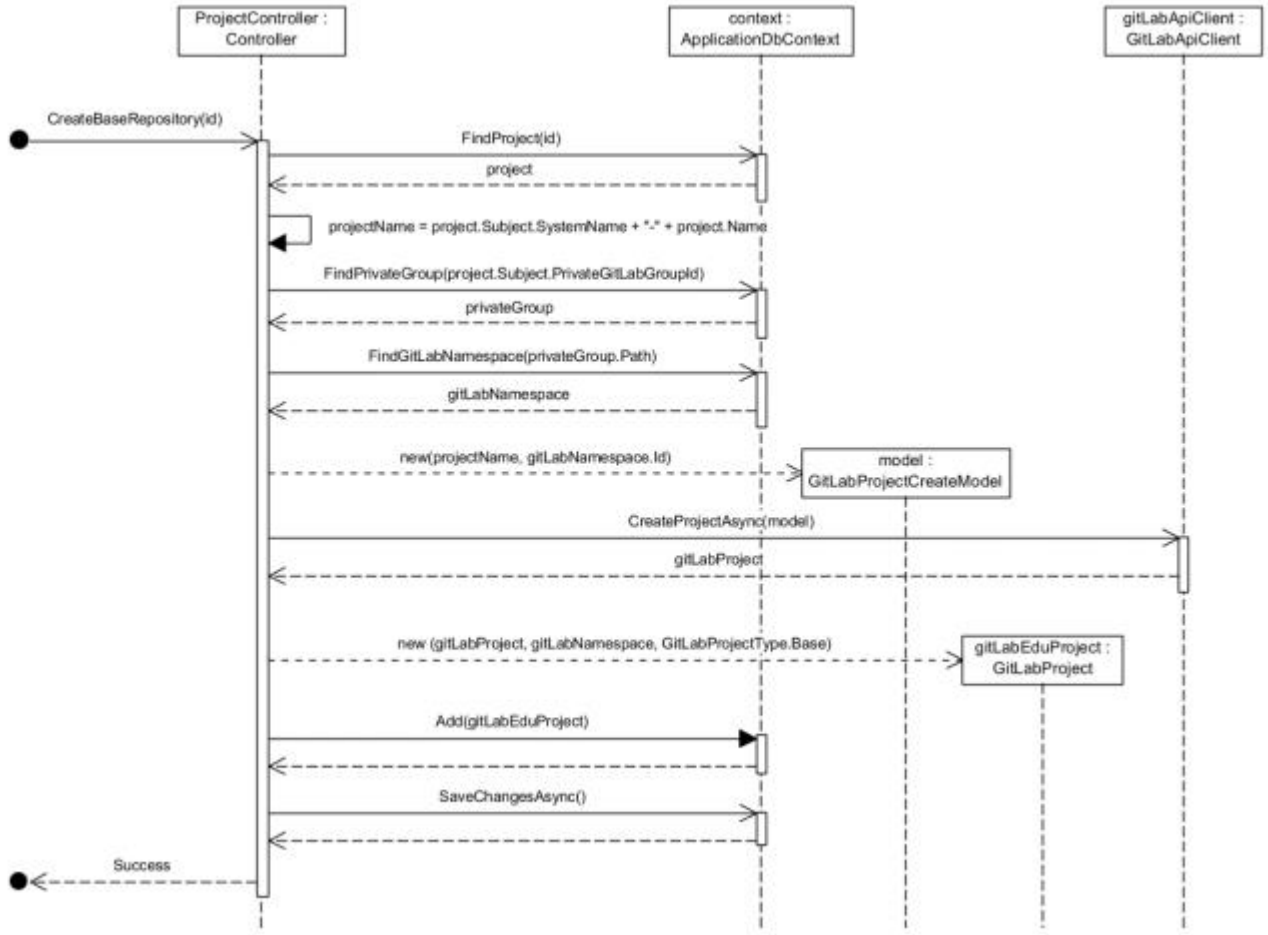

Рисунок 2.13 – Створення базового репозиторію проекту

На діаграмі послідовності на рис. 2.14 показано процес створення репозиторію, призначеного для авторського рішення Вклд. Даний тип репозиторію доступний тільки в проектах типу «Домашня робота» та «Контрольна робота», а також у їх командних аналогах. Репозиторій створюється на основі базового репозиторію проекту в приватній групі Пр та доступний лише Вклд. Так як GitLab не дозволяє виконати операцію Fork для створення проекту в тому ж просторі імен з іншим ім'ям, ця операція емулюється шляхом створення порожнього нового проекту і перенесення коммітів з базового репозиторію шляхом клонування базового репозиторію в тимчасовий каталог сервера (git clone) та подальшого відправлення змін у щойно створений порожній репозиторій для авторського рішення (git push).

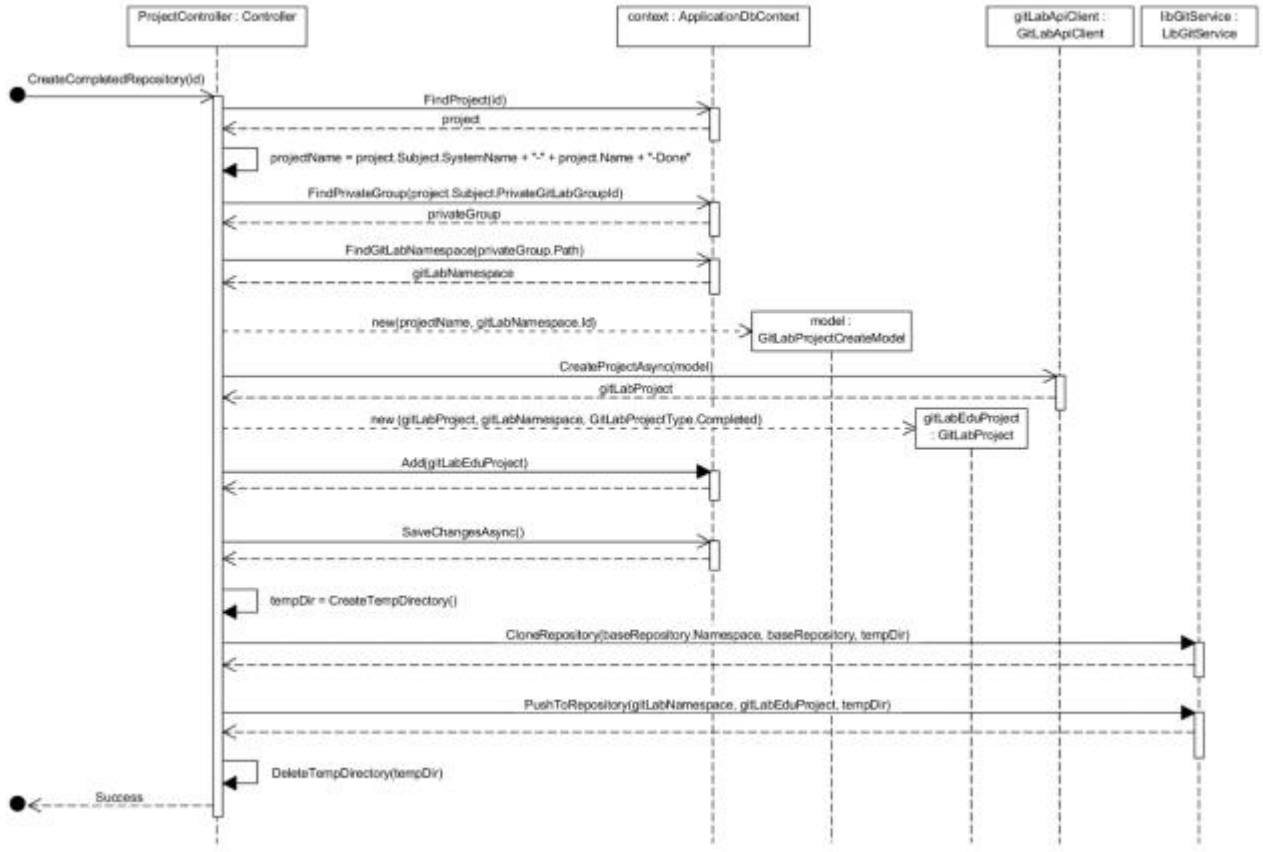

Рисунок 2.14 – Створення репозиторію для авторського рішення

На діаграмі послідовності, наведеної на рис. 2.15? зображений процес копіювання базового репозиторію командам. Дана операція застосовна тільки для проектів типу «Командна домашня робота» та «Командна контрольна робота» (тільки для командних проектів) і дозволяє Вклд скопіювати шаблонний базовий репозиторій всім командам (у разі з командною роботою, надалі, Вклд може відправити командам конкретні варіанти, шляхом надсилання їм додаткових коммітів з репозиторіїв, створених для варіантів). У зв'язку з тим, що репозиторії різних команд створюються в одній групі (командній групі Пр), дана операція не можемо використовувати метод Fork Project, і реалізована шляхом створення порожнього репозиторію в командній групі та перенесення коммітів через операції git clone і git push.

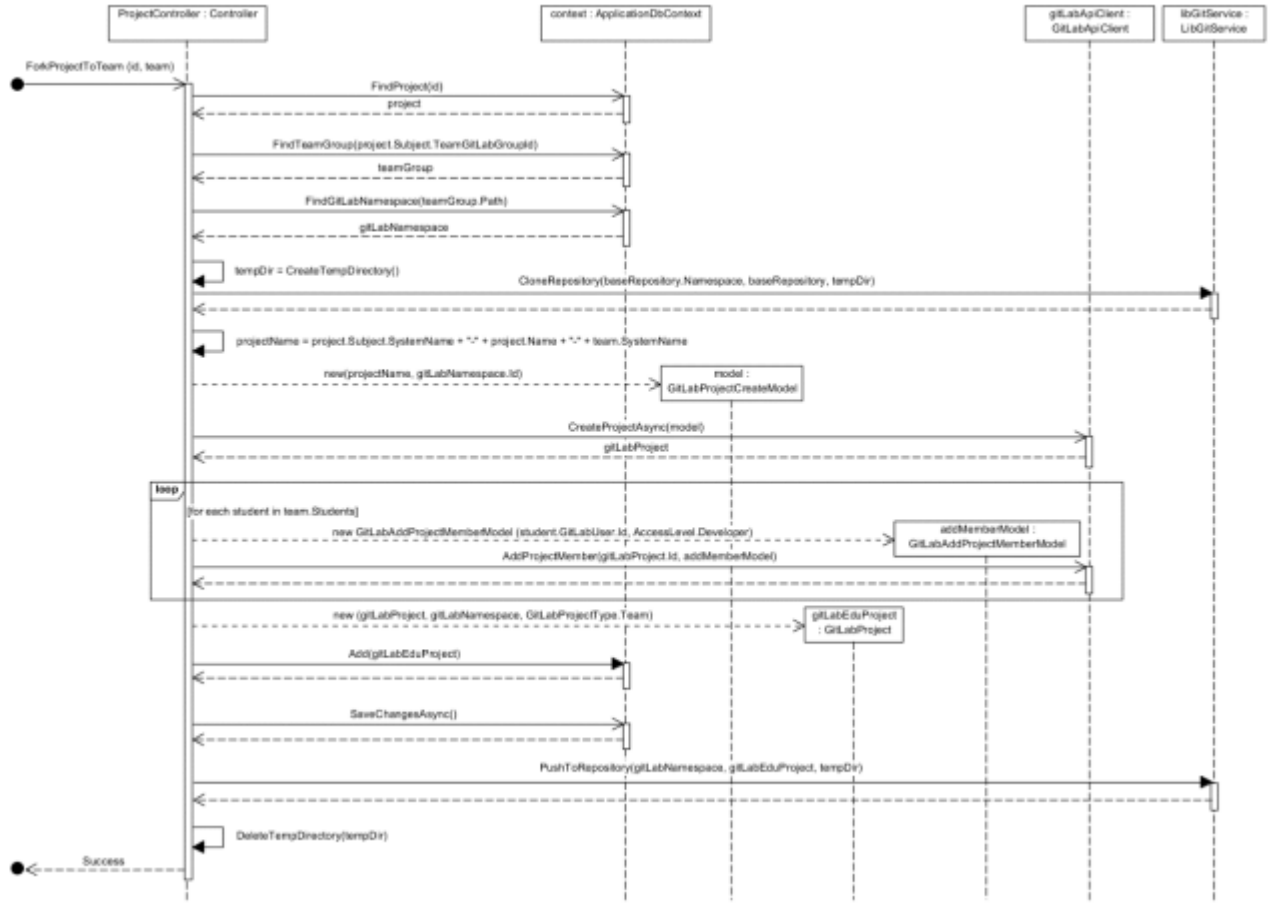

Рисунок 2.14 – Копіювання базового репозиторію командам

2.8 Взаємодія з Git -репозиторіями

Для виконання деяких дій, котрі зв’язані з керуванням проектами, необхідно, окрім застосування API GitLab, працювати з Git -репозиторіями безпосередньо, наприклад для копіювання репозиторіїв усередині одного простору імен, або для завантаження завдання контрольної роботи в існуючі репозиторії Стд.

Для цього, в системі використовується бібліотека libgit2sharp, котра і реалізує необхідні методи по взаємодії з Git -репозиторіями і дозволяє розроблюваній системі виконувати їх клонування, а також відправляти зміни з одного репозиторію в інший.

Для полегшення використання цієї бібліотеки було використано патерн «Фасад». Було створено службу LibGitService (рис.2 16), що містить такі методи:

– CloneRepository - здійснює клонування зазначеного віддаленого

репозиторію у вказану локальну папку (операція, еквівалентна команді git clone);

– PushToRepository - проводить надсилання змін з локального репозиторію із зазначеної папки до зазначеного віддаленого репозиторію (операція, еквівалентна команді git push).

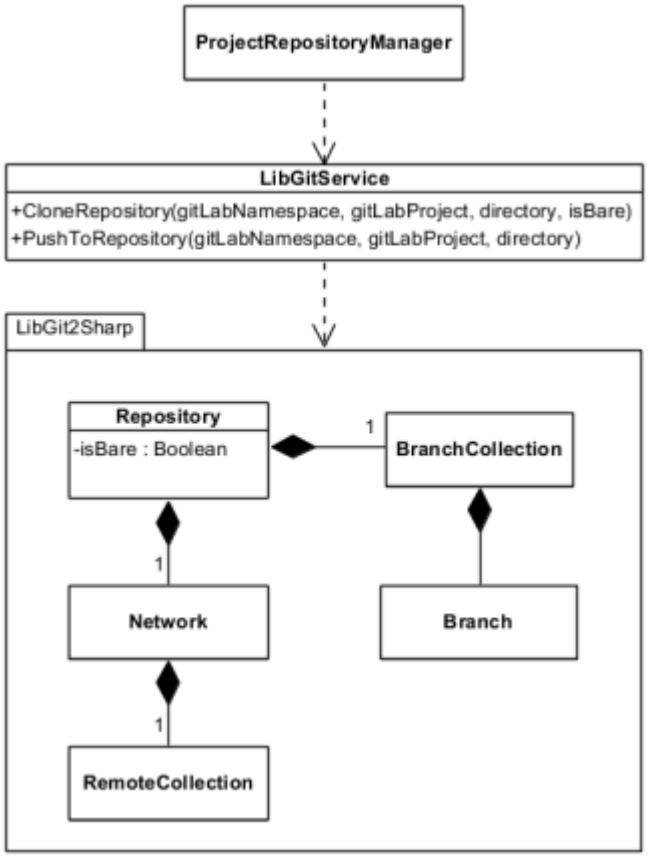

Рисунок 2.16 – Служба LibGitService

РОЗДІЛ 3. ПРАКТИЧНА ЧАСТИНА

3.1 Архівація та збирання коду проектів

При використанні системи протягом кількох років, у ній будуть накопичуватися дані різних Стд, у тому числі тих, які вже не є учасниками Пр або взагалі вже завершили навчання в університеті Для того, щоб не захаращувати інтерфейс Вклд неактуальними Стд (а також командами та проектами), в системі передбачено механізм архівації, який з одного боку дозволяє зберегти Стд, команди та проекти, а з іншого - приховує їх від Вклд.

Архівація Стд. Викладач може виконати архівацію будь-якого Стд, записаного на його Пр. В результаті архівації, у Стд архівуються його репозиторії - Стд втрачає можливість вносити в них зміни, але зберігає доступ до них. Також, заархівований Стд не відображатиметься Вклд у списках вибору Стд.

У випадку, якщо Стд потрібно пройти курс повторно, він може бути розархівований Вклд.

Архівація репозиторіїв. Архівація репозиторіїв (рис. 3.1) може бути необхідна в тих випадках, коли Стд або команді необхідно повторно скопіювати репозиторій (наприклад, для передачі контрольної роботи або повторного проходження курсу). В результаті архівації репозиторій перейменовується: йому додається рік і послідовний номер, якщо виявилося недостатньо для збереження унікальності. Архівація репозиторіїв є незворотною дією.

Архівація команд. Архівація команд дозволяє зберегти команди та їх учасників для історії, але при цьому дозволяє повторно створювати команди з такими ж іменами з такими ж або іншими учасниками. При архівації команд виконується архівація репозиторіїв. Архівація команди є незворотною дією.

Блокування репозиторіїв. Крім незворотної архівації репозиторіїв, Вклд може зробити тимчасове їх блокування (наприклад, для того щоб Стд не могли вносити зміни до контрольної роботи після її закінчення), яке потім може бути знято.

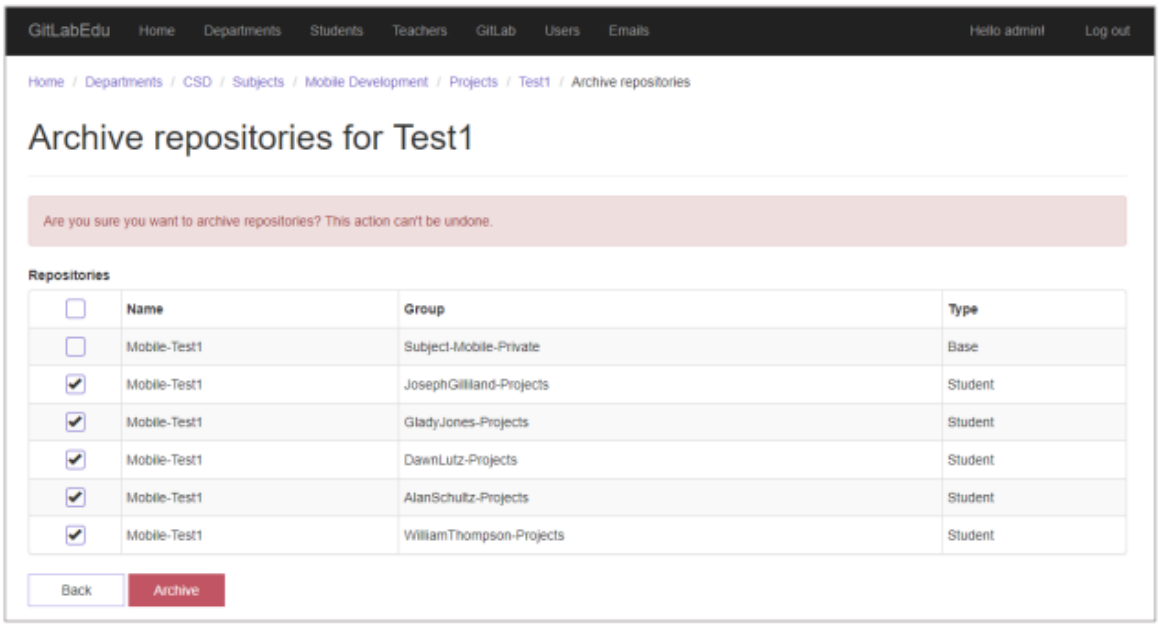

Рисунок 3.1 – Архівація репозиторіїв проекту

Збір коду проекту. Така операція (рис. 3.2) дозволяє Вклд швидко зібрати репозиторії Стд або команд для перевірки. Для цього система клонує їх у свою тимчасову директорію, пакує у zip-архів, який потім може бути завантажений Вклд.

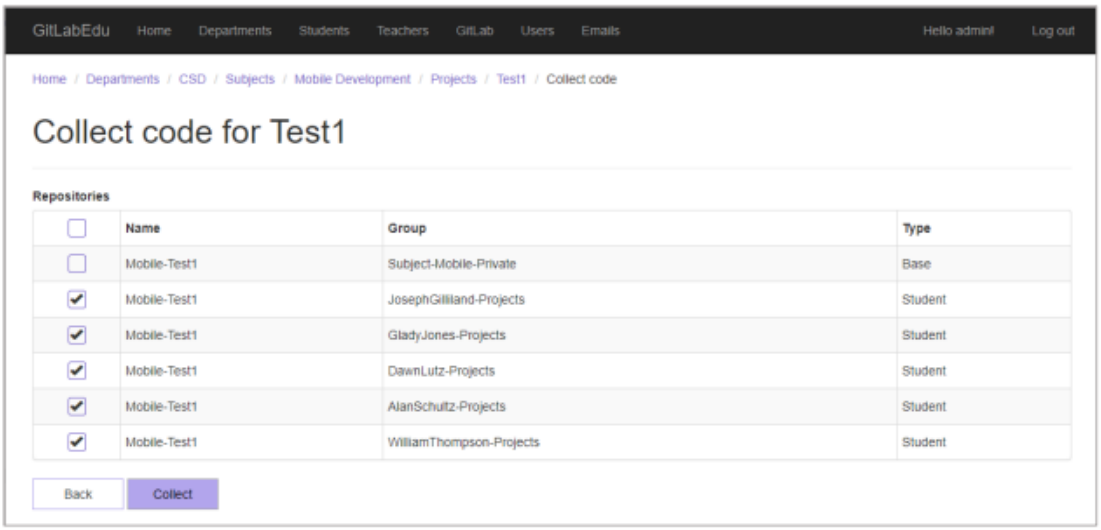

Рисунок 3.2 – Збір коду проекту

3.2 Імпорт студентів

Для того, щоб полегшити додавання до системи великої кількості Стд

одноразово (додавання групи або потоку Стд), реалізовано механізм імпорту.

Під час імпорту Адм пропонується вибрати факультет і групу Стд, що імпортуються, а також додати список Стд, котрих необхідно імпортувати (рис. 3.3). При виконанні імпорту Адм буде також запропоновано створити користувача та особисту групу GitLab для кожного Стд (рис. 3.4).

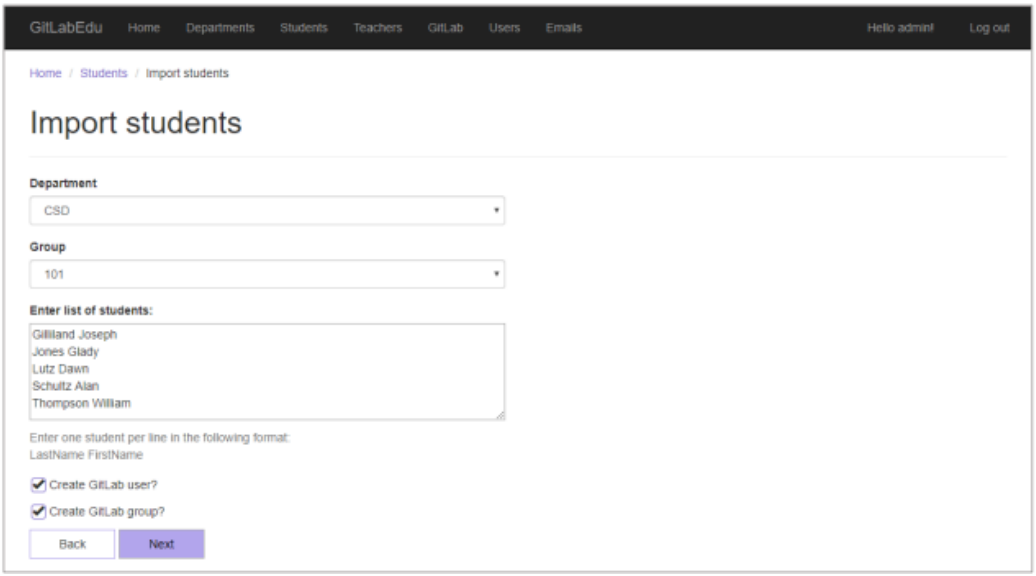

Рисунок 3.3 – Стартова сторінка імпорту Стд

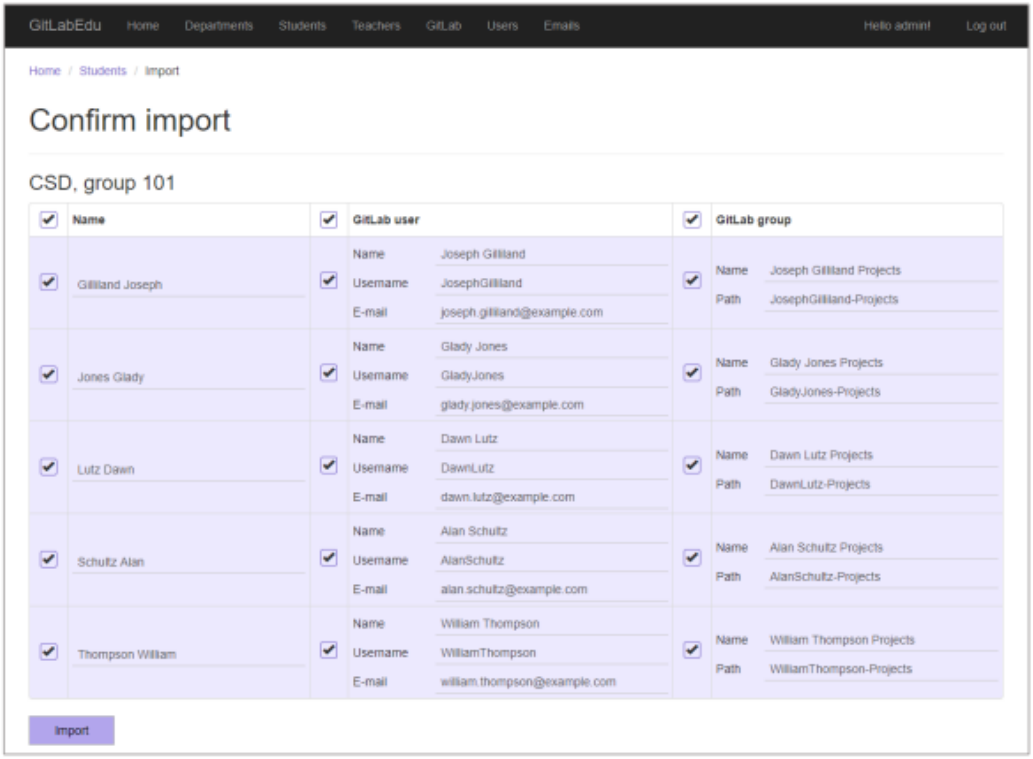

Рисунок 3.4 – Сторінка підтвердження імпорту Стд

Після виконання імпорту система формує звіт про імпорт (рис. 3.5), а також дозволяє роздрукувати аркуші з паролями (рис. 3.6), які потім можна роздати Стд.

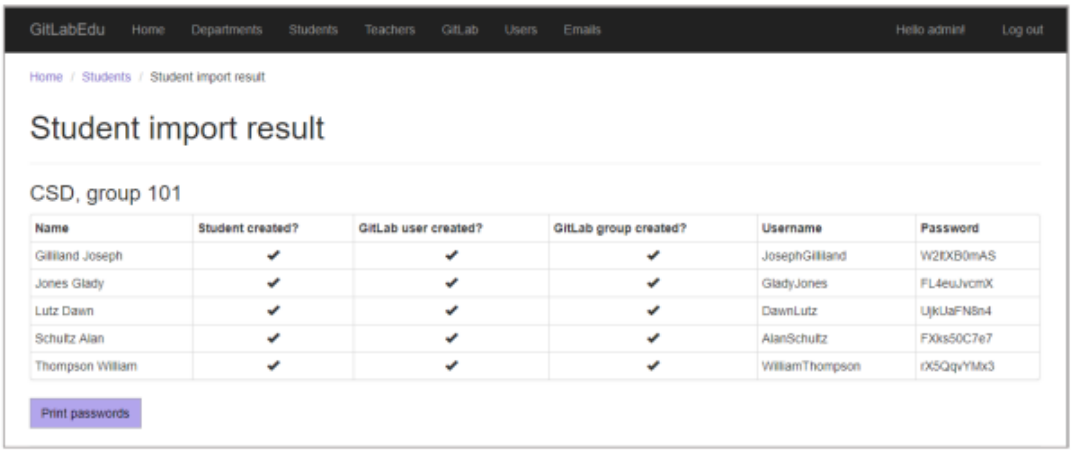

Рисунок 3.5 – Сторінка результату імпорту Стд

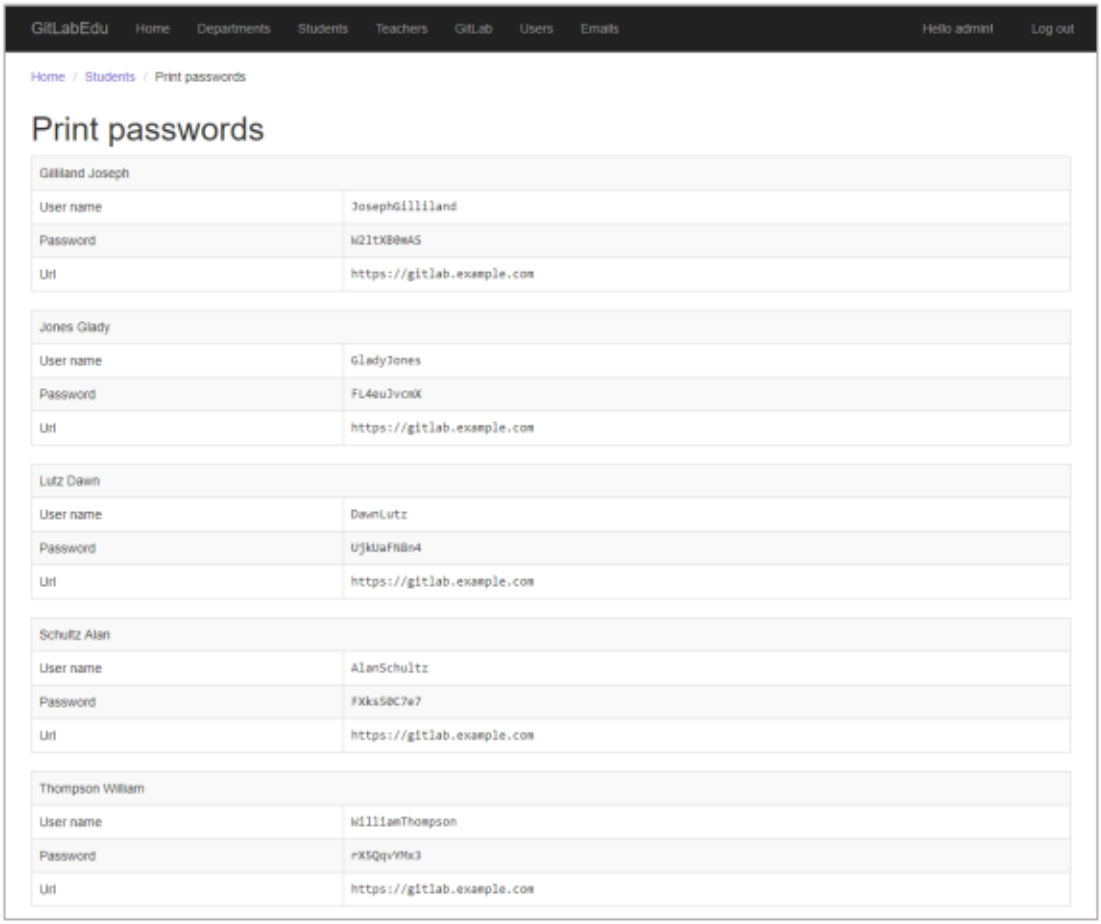

Рисунок 3.6 – Сторінка з паролями Стд

3.3 Статистика

Для відстеження прогресу Стд додано можливість переглядати статистику їх роботи. Статистичні дані збираються з кожного Пр в розрізі Стд та проектів. У Вклд у межах Пр є можливість створювати шаблони звітів. Шаблони звітів дозволяють вибрати Стд та проекти, для яких необхідно будувати звіти, а також задати тип агрегованих значень: кількість коммітів у репозиторії проектів або сумарний розмір змін у них. У подальшому, на підставі шаблону може бути побудований безпосередньо сам звіт у вигляді таблиці, стовпці якої є проектами, рядки - Стд, а на їхньому перетині відображається їх результат (рис. 3.7).

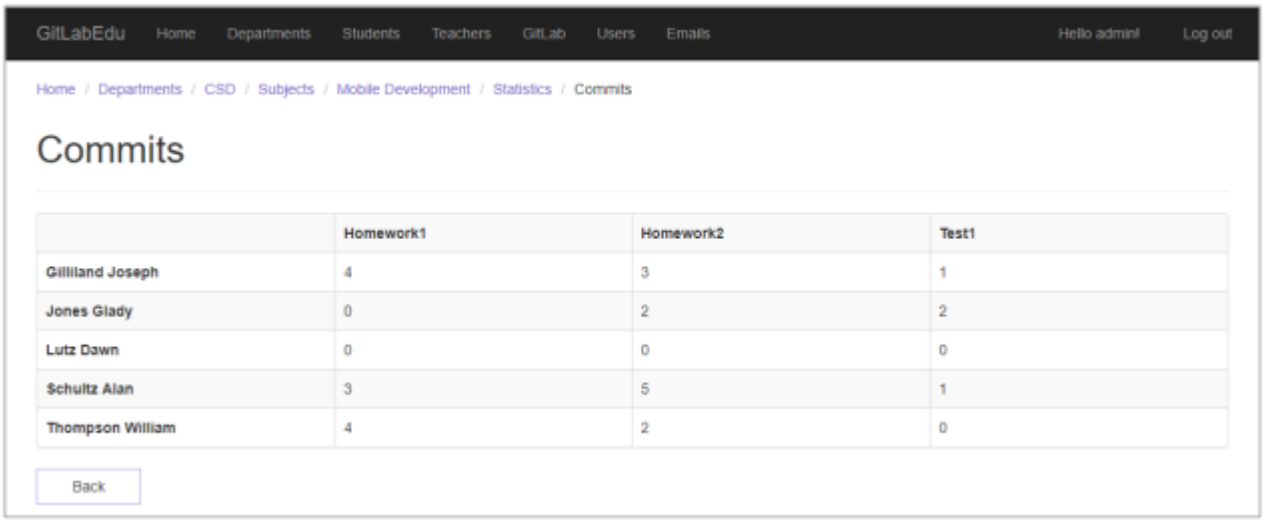

Рисунок 3.7 – Статистика коммітів

3.4 Розгортання системи

3.4.1 Способи розгортання

Різні компоненти системи мають різні вимоги до розгортання:

- GitLab може бути розгорнутий тільки на UNIX -подібній ОС;
- розроблюваний додаток, є кросплатформним і може бути розгорнутий як на UNIX –подібній, так і на Windows ОС;
- MS SQL Server також є кросплатформовою і може бути розгорнуто як на Windows, так і на UNIX -подібній ОС.

GitLab постачається у вигляді Omnibus -пакета [10], що включає всі необхідні компоненти для роботи GitLab, включаючи вбудований ВС nginx і СУБД Post-greSQL. При установці GitLab на окремому сервері, достатньо встановити даний Omnibus -пакет і виконати його налаштування (шляхом редагування файлу /etc/gitlab/gitlab.rb).

Для коректної роботи засобу, що розробляється, необхідно застосовувати ВС. При розгортанні додатку на ОС Windows, як ВС, зазвичай виступає, IIS, що постачається разом з ОС. Для цього треба інсталювати ASP.NET Core/.NET Core Runtime & Hosting Bundle, що включає ASP.NET Core IIS Module, необхідний для підтримки ASP.NET Core -додатків і веб-сервері IIS.

При розгортанні на UNIX -подібній ОС можна застосувати ВС nginx. Цей ВС буде зворотнім проксі до Kestrel - вбудованого ВС додатку та перенаправляти йому всі запити до нього. При цьому необхідно забезпечити запуск самої веб-програми (наприклад, у вигляді служби systemd).

Іншим підходом до розгортання системи є використання платформи контейнеризації Docker, при використанні якої система може бути розгорнута у вигляді набору Кн, що відповідають за різні її компоненти (GitLab, MS SQL, розробляється ASP.NET Core -додаток).

3.4.2 Платформа контейнеризації Docker

Docker — ПЗ для автоматизації розгортання та керування програмами в середовищах з контейнеризацією [11]. Контейнеризація - це спосіб поділу програмних додатків один від одного за рахунок приміщення кожного додатка з повним оточенням його та параметрами у Кн.

За допомогою платформи Docker користувач може розгортати свої програми в ізольованому програмному середовищі (Кн) для запуску на ОС хоста. Основна перевага Docker –дозволяє користувачам упакувати додаток з усіма його залежностями в стандартизований блок для розробки ПЗ.

Кожен Кн виконується як окремий процес в ОС Linux та поділяє ядро хост-машини з різними Кн. Таким чином, один запущений Кн займає не більше пам'яті, ніж який-небудь інший виконуваний файл, що робить його легковажним.

Кн не вимагають великих накладних витрат, забезпечують ефективніше використання ОС та ресурсів.

Docker містить три частини:

– ПЗ. Демон Docker є процесом, котрий керує Кн Docker і опрацьовує об'єкти Кн. Він прослуховує запити, котрі надіслані власне через API Docker Engine.

– Об'єкти. Застосовуються для монтування програми в Docker. Головними видами об'єктів Docker є образи, Кн та сервіси.

– Реєстри. Це сховище образів Docker. Властиво клієнти Docker під'єднуються до них, щоб «pull» зображення для застосування або «push» створені ними зображення.

Для того, щоб описувати та виконувати програми, що складаються з набору Кн, зв'язаних між собою, Docker надає інструмент docker-compose [12]. Конфігураційні файли docker-compose описуються мовою YAML, де відбувається налаштування різного роду програмних служб і виконується процес формування та викоання всіх Кн за використання єдиної команди.

Робота з Docker у найпростішому випадку, якщо необхідний образ вже було зібрано і є у публічному реєстрі, відбувається так:

- інсталювати потрібну версію Docker на платформі, що підтримується;
- здійснити завантаження готовий образу з реєстру (docker pull);
- запустити docker- Кн з урахуванням цього образу (docker run).

Якщо потрібного готового рішення немає у реєстрі, можна зібрати свій образ. Для того щоб створити свій Кн існує механізм складання — docker build, який використовує набір інструкцій з Dockerfile для складання нового образу.

У системі, що розробляється, взаємодіятимуть один з одним чотири окремі компоненти:

– ВС nginx;

– GitLab;

– сама програма;

– БД MS SQL.

Так як в Docker для віртуалізації одного процесу прийнято створювати

окремий Кн, було вирішено створити чотири Кн, кожен з яких запускатиме потрібний компонент системи.

3.4.3 Створення образів Docker

Це відбувається автоматично, із використанням вказівок з Dockerfile – текстового файлу, який має усі вказівки в порядку, необхідному для створення даного зображення. Образ містить шари, кожен із яких представляє інструкцію Dockerfile. Вони накладаються один на одного, і кожен із них є дельтою змін попереднього. За підсумками всі шари накладаються на вихідний образ (образ ОС) і виходить готовий результат.

Додатки, що складаються з декількох Кн, можна застосовувати інструментом docker-compose, котрийдає описує систему як набір серверів, при допомозі файлу конфігурації docker-compose.yml. Він призначений для визначення сервісів програми і містить набір атребутів конфігурації: використовуваний образ, шлях до Dockerfile, змінні оточення, перенаправлення - портів, використовувані томи.

У системі, що розробляється, будуть використані наступні сервіси:

– mssql - MS SQL для зберігання даних.

Для цього сервісу буде застосовано офіційний образ SQL Server , який завантажується з реєстру MS Container Registry: mcr.microsoft.com/mssql/server

З метою коректної роботи сервісу необхідно налаштувати MS SQL за допомогою наступних змінних оточення:

– MSSQL_PID – визначає редакцію MS SQL. Для системи, що розробляється, достатньо редакції MS SQL Server Express Edition;

– ACCEPT_EULA - підтвердження ухвалення ліцензійної угоди MS SQL, необхідно встановити рівним Y;

– SA_PASSWORD – пароль для облікового запису Адм SA.

Для забезпечення збереження даних, необхідно примонтувати каталог хоста (bind mount) або том даних (volume) для наступного каталогу Кн:
/var/opt/mssql - містить усі БД MS SQL.

– gitlab – власне система GitLab, яка надаватиме Стд і Вклд доступ до Git

-репозиторію

Для даного сервісу буде використаний офіційний образ GitLab CE, який завантажується з реєстру Docker Hub: gitlab/gitlab-ce.

Для коректної роботи даного сервісу треба налаштувати GitLab із застосуванням змінної оточення GITLAB_OMNIBUS_CONFIG.

Для забезпечення збереження даних необхідно примонтувати наступні каталоги Кн:

– /etc/gitlab - містить налаштування GitLab;

– /var/opt/gitlab - містить дані GitLab, наприклад Git -репозиторії або файли БД PostgreSQL, використовуваної GitLab;

– /var/log/gitlab – містить файли журналу різних підсистем GitLab.

– manager - система, що розробляється, виконана у вигляді ASP.NET Core - додатка. Образ для цього сервісу буде збиратися на основі Dockerfile.

Для коректної роботи даного сервісу необхідно виконати налаштування ASP.NET Core за допомогою наступних змінних оточення:

– ASPNETCORE_ENVIRONMENT – тип оточення, наприклад Production.

– ASPNETCORE_URLS - набір адрес, за якими має бути доступний ASP.NET Core –додаток, який запускається , наприклад http://+:5000.

Для забезпечення збереження ключів програми, а також для того, щоб не захаращувати шар КН тимчасовими файлами, необхідними для роботи програми (наприклад, тимчасово клонованими репозиторіями), необхідно примонтувати наступні каталоги Кн:

– /app/keys - містить ключі шифрування, що використовуються в тому числі для захисту аутентифікаційних токенів. Якщо не зберігати цей каталог, то при кожному перезапуску Кн всі користувачі будуть автоматично розлогінені;

– /app/tmp - містить тимчасові дані програми, наприклад тимчасово клоновані репозиторії.

– nginx - ВС nginx, який виступає як зворотний проксі і перенаправляє - HTTP-запити до GitLab і ASP.NET Core - додатку .

Для цього сервісу буде використано офіційний образ nginx, що завантажується з реєстру Docker Hub.

Для коректної роботи даного сервісу необхідно примонтувати наступний - каталог Кн: /etc/nginx/conf.d - в даному каталозі необхідно розмістити конфігураційні файли для віртуальних серверів, що перенаправляють запити до GitLab і ASP.NET Core додатку.

Також необхідно виконати перенаправлення наступних портів:

– 80 - стандартний порт HTTP;

– 443 - стандартний порт HTTPS (вимагає додаткової установки та - установки сертифікатів).

На рис. 3.8 представлена діаграма розгортання даної системи

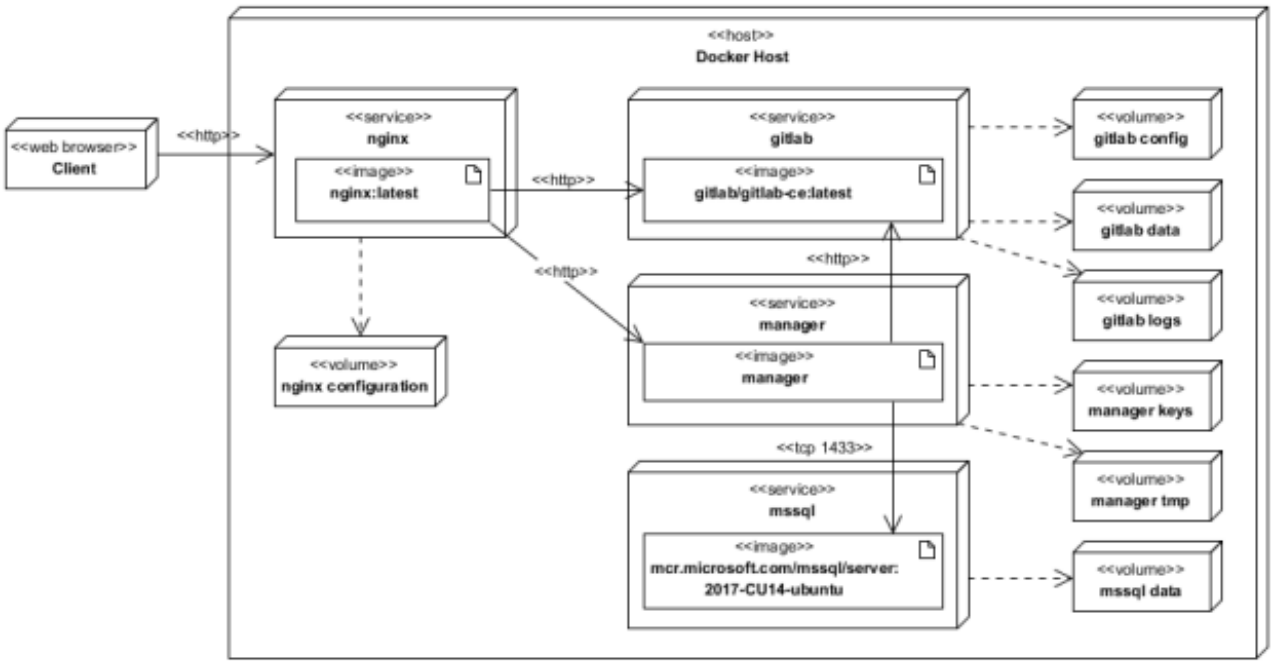

Рисунок 3.8 – Діаграма розгортання системи, котра розробляється

У лістингу 3.1 наведено файл docker-compose.yml за допомогою якого може бути запущена система.

Лістинг 3.1 - Приклад файла docker-compose.yml (фрагмент)

```
version: '3'
services:
  gitlab:
    image: 'gitlab/gitlab-ce:latest'
    environment:
      GITLAB_OMNIBUS_CONFIG: |
        # Set to actual external_url
        external_url 'http://gitlab.example.com'
        nginx['listen_addresses'] = ['*']
        nginx['listen_port'] = 80
        nginx['listen_https'] = false
        # Set Host to actual external_url
        nginx['proxy_set_headers'] = {
          "Host" => "gitlab.example.com",
          "X-Forwarded-Proto" => "http",
        }
        # Email config
        gitlab_rails['gitlab_email_enabled'] = false
        gitlab_rails['incoming_email_enabled'] = false
        # Other settings
        gitlab_rails['gitlab_default_can_create_group'] = false
        gitlab_rails['gitlab_username_changing_enabled'] = false
        # Add any other gitlab.rb configuration here, each on its own line
    volumes:
      - './data/gitlab/config:/etc/gitlab'
      - './data/gitlab/logs:/var/log/gitlab'
```

Лістинг 3.2 містить програмний код Dockerfile для складання образу для сервісу manager (ASP.NET Core -додаток системи, що розробляється),

Лістинг 3.2 - Dockerfile, котрий використовується для зборки образу служби manage

```
FROM mcr.microsoft.com/dotnet/core/aspnet:2.2
WORKDIR /app
COPY ./app .
ENTRYPOINT ["dotnet", "GitLabEdu.dll"]
```

4 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Навчання працюючих і інструктажі з охорони праці

Однією із складових ефективної роботи з профілактики виробничого травматизму є належна підготовка, навчання та підвищення кваліфікації працівників з питань охорони праці. Загальний порядок проведення навчання з питань охорони праці встановлений Законом України «Про охорону праці» (ст. 18. «Навчання з питань охорони праці»).

Виконання вимог Закону України «Про охорону праці» в частині проведення навчання та перевірки знань з питань охорони праці здійснюється відповідно до Типового положення про порядок проведення навчання і перевірки знань з питань охорони праці, затвердженого наказом Держкомітету України з нагляду за охороною праці 26 січня 2005 р. № 15 (далі — Типове положення).

Нагляд за дотриманням вимог Типового положення здійснюють органи державного нагляду за охороною праці, а координацію та методичний супровід — Головний навчально-методичний центр та навчальні підрозділи експертно-технічних центрів Держгірпромнагляду [13].

Вивчення предмета «Охорона праці» при підготовці, перепідготовці та підвищенні кваліфікації працівників, які залучаються до виконання робіт з підвищеною небезпекою, на підприємстві регламентується п. 2.3 Типового положення. На підприємствах згідно з п. 1.1 Додатку 3 Типового положення навчання та перевірку знань з питань охорони праці повинні проходити керівники, заступники керівників, головні спеціалісти, керівники основних виробничих та технічних служб, безпосередньо пов’язані з організацією безпечного ведення робіт. Крім цього, згідно з п. 5 Додатку 3, навчання та перевірку знань з питань охорони праці мають проходити керівники, спеціалісти служб охорони праці, члени комісій з перевірки знань з питань охорони праці, особи, відповідальні за технічний стан і безпечну експлуатацію об’єктів підвищеної небезпеки підприємств [14].

Типове положення встановлює порядок та місце проведення навчання та перевірки знань з питань охорони праці посадових осіб (п. 5.2 та п. 5.3). Посадові особи, перелік яких наведено в п. 5.2, проходять навчання у Головному навчально-методичному центрі Держнаглядохоронпраці. Перевірка знань цієї категорії посадових осіб проводиться комісією, створеною наказом Держгірпромнагляду.

Організацію навчання та перевірки знань з питань охорони праці працівників на підприємстві здійснюють працівники служби кадрів або інші спеціалісти, яким роботодавець доручив організацію цієї роботи. Навчання та перевірка знань з питань охорони праці працівників (виконавців і посадових осіб), які не залучаються до виконання робіт підвищеної небезпеки, проводиться не рідше ніж один раз на три роки. Посадові особи та працівники, які виконують роботи підвищеної небезпеки, проходять спеціальне навчання та перевірку знань відповідних нормативно-правових актів з охорони праці не рідше одного разу на рік.

Посадові особи малих підприємств (п. 5.4), які не мають можливості створити власні комісії з перевірки знань з питань охорони праці та провести навчання з питань охорони праці, проходять навчання та перевірку знань в навчальних закладах, які мають відповідний дозвіл.

Спеціальне навчання з питань охорони праці може проводитись безпосередньо на підприємстві або навчальним закладом, який має відповідний дозвіл. При проведенні такого навчання на підприємстві навчальні плани та програми розробляються з урахуванням конкретних видів робіт, виробничих умов і функціональних обов’язків працівників і затверджуються наказом керівника підприємства [14].

Періодичність інструктажів, навчання та перевірки знань з питань охорони праці залежить від видів виконуваних робіт та встановлюється Типовим положенням. Перевірка знань з питань охорони праці після проведення спеціального навчання проводиться комісією підприємства.

Якщо на підприємстві неможливо створити комісію з перевірки знань з питань охорони праці (п. 4.4 Типового положення), перевірка знань проводиться

комісією спорідненого підприємства або Теруправління Держгірпромнагляду.

Всі працівники та посадові особи підприємства, включаючи посадових осіб, відповідальних за виконання робіт підвищеної небезпеки (крім зазначених в п. 5.2 та п. 5.3 Типового положення), проходять навчання та перевірку знань з питань охорони праці на підприємстві. Типове положення не зобов'язує, але й не забороняє проводити навчання всіх виконавців та посадових осіб (особливо тих, що виконують роботи підвищеної небезпеки) в навчальних закладах. У нашій країні є багато підприємств, де таке навчання проводиться, і це має позитивні наслідки. Ті витрати, які при цьому несуть підприємства, перекриваються створенням більш безпечних умов праці і в результаті збереженням життя та здоров'я працівників.

Також в навчальних закладах проходять навчання та перевірку знань із загальних питань охорони праці всі посадові особи та фахівці, які проводять інструктажі або навчання підлеглих працівників з питань охорони праці, виконують роботи з проектування об'єктів, а також інші працівники, незалежно від того, передбачено таке навчання Типовим положенням чи ні.

4.2 Санітарно-гігієнічні вимоги до умов праці

На робочих місцях працівників, які відповідальні за експлуатацію сервісу управління механізмом авторських прав на мультимедійні файли, необхідно забезпечити дотримання вимог, затверджених Наказом Мінсоцполітики від 14.02.2018 за № 207 «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями».

Приміщення для роботи з ЕОМ мають бути обладнані системами опалення, кондиціонування повітря, або припливно-витяжною вентиляцією. У приміщеннях на робочих місцях мають забезпечуватись оптимальні значення параметрів мікроклімату: температури, відносної вологості та рухливості повітря відповідно до норм та правил, а також ДБН В.2.5-67:2013 «Опалення, вентиляція та кондиціонування», затверджених наказом Мінрегіону від 25.01.2013 р. № 24. Відповідно до санітарних норм мікроклімату виробничих приміщень ДСН

3.3.6.042-99 в офісних приміщеннях, обладнаних ЕОМ, температура повітря повинна становити 22-25°С, відносна вологість повітря – 40-60 %, швидкість руху повітря – не більше 0,1 м/с [14].

Приміщення, призначені для роботи з ЕОМ, повинні мати природне освітлення. У виробничих приміщеннях, обладнаних ЕОМ, необхідно створити належне освітлення. При експлуатації сервісу управління механізмом авторських прав на мультимедійні файли, важливим, з точки зору охорони праці, є забезпечення достатньої величини природного та штучного освітлення, які визначені у НПАОП 0.00-7.15-18 [15]. Природне світло повинно бути бічним, зорієнтованим, як правило, на північ чи північний схід, і забезпечувати коефіцієнт природної освітленості не нижче 1,5%. При виробничій потребі дозволяється експлуатувати ЕОМ у приміщеннях без природного освітлення за узгодженням з органами Держпромгірнагляду та органами й установами санітарно-епідеміологічної служби. Вікна приміщень повинні мати регулювальні пристрої для відчинення, а також жалюзі, штори тощо. Штучне освітлення приміщення з робочими місцями, обладнаними відеотерміналами ЕОМ загального та персонального користування, має бути всеосяжним і рівномірним. У випадку, коли переважають роботи з документами, допускається комбіноване освітлення (додатково до загального освітлення встановлюється світильники місцевого освітлення). Світильники розміщуються збоку від робочих місць (переважно ліворуч), або локально над робочим місцем (при розташуванні відеотерміналів ЕОМ за периметром приміщення). Як джерело світла при штучному освітленні застосовуються, як правило, люмінесцентні лампи. У світильниках місцевого освітлення допускається застосування ламп розжарювання. Рівень освітленості на робочому місці має становити 300-500 лк. При використанні комбінованого освітлення не допускається відблисків на поверхні екрана та збільшення освітлення екрана вище 300 лк.

Орієнтація вікон повинна бути на північ або північний схід, вікна повинні мати жалюзі, які можна регулювати, або штори; не дозволяється розміщувати кабінети обчислювальної техніки у підвальних приміщеннях будинків; кабінети, обладнані комп'ютерною технікою, в навчальних закладах повинні

розміщуватись в окремих приміщеннях з природним освітленням та організованим обміном повітря; стіни, стеля і підлога та обладнання кабінетів комп'ютерної техніки повинні мати покриття із матеріалів з матовою фактурою з коефіцієнтом відбиття: стін — 40- 50 %, стелі — 70 - 80 %, підлоги — 20-30 %, предметів обладнання — 40-60 % (робочого столу — 40-50 %, корпуса дисплею та клавіатури — 30-50 %, стелажів — 40-60 %); поверхня підлоги повинна мати антистатичне покриття та бути зручною для вологого прибирання; забороняється використовувати для оздоблення інтер'єру приміщень комп'ютерних кабінетів полімерні матеріали (дерев'яно-стружкові плити, шпалери, що придатні для миття, плівкові та рулонні синтетичні матеріали, шаровий паперовий пластик та ін.), що виділяють у повітря шкідливі хімічні речовини, які перевищують гранично допустимі концентрації; вміст шкідливих хімічних речовин в повітрі дошкільних та учбових приміщень з комп'ютерною технікою не повинен перевищувати середньодобові концентрації [15].

Організація робочого місця фахівця із експлуатації сервісу повинна забезпечувати відповідність усіх елементів робочого місця та їх розташування ергономічним вимогам ДСТУ 8604:2015 «Дизайн і ергономіка. Робоче місце для виконання робіт у положенні сидячи. Загальні ергономічні вимоги».

Відстань від екрана до ока фахівців, які працюють за комп’ютером визначається згідно з вимогами ДСанПіН 3.3.2.007-98.

Рівень шуму не повинний перевищувати: на місцях, де працюють програмісти та оператори ЕОМ, 55 дБА, у лабораторіях, де складаються алгоритми та ведеться робота з документацією – 60 дБА, у машинному залі – 65 дБА, у приміщеннях, де розміщені гучні агрегати обчислювальних машин – 75 дБА.

ВИСНОВКИ

У рамках кваліфікаційної роботи було розроблено та впроваджено веб-додаток для автоматизації управління Git -репозиторіями на базі системи GitLab для використання в процесі навчання.

Ця програма використовується Вклд для організації процесу навчання Стд у системі управління репозиторіями GitLab. Ця система застосовується для видачі завдань Стд, проведення контрольних робіт, публікації матеріалів по Пр, а також надає Стд простір для розміщення своїх нотаток у особистих репозиторіях.

Крім допомоги Вклд, розробка використовується для мінімізації роботи Адм у рамках системи реалізована можливість керування Стд, Вклд та Пр з можливістю автоматичного формування користувачів та груп у GitLab для них.

ПЕРЕЛІК ДЖЕРЕЛ

1. Git. [Електронний ресурс]. – Режим доступу: https://git-scm.com (дата звертання: 10.04.2022).
2. The only single product for the complete DevOps lifecycle – GitLab. [Електронний ресурс]. – Режим доступу: https://about.gitlab.com (дата звертання: 10.04.2022).
3. Introduction to ASP.NET Core. [Електронний ресурс]. – Режим доступу: https://docs.microsoft.com/en-us/aspnet/core/ (дата звертання: 10.04.2022).
4. Martin Fowler. Patterns of Enterprise Application Architecture. – Addison-Wesley Professional, 2002. – 560 p.
5. Entity Framework Core Quick Overview. [Електронний ресурс]. – Режим доступу: https://docs.microsoft.com/en-us/ef/core/ (дата звертання: 10.04.2022).
6. Bootstrap – The world's most popular mobile-first and responsive front-end framework. [Електронний ресурс]. – Режим доступу: https://getbootstrap.com/docs/3.3/ (дата звертання: 10.04.2022).
7. LibGit2Sharp. [Електронний ресурс]. – Режим доступу: https://github.com/libgit2/libgit2sharp (дата звертання: 10.04.2022).
8. Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides. Design Patterns: Elements of Reusable Object-Oriented Software. – Addison-Wesley Professional, 1994. – 416 p.
9. GitLab API. [Електронний ресурс]. – Режим доступу: https://docs.gitlab.com/ee/api/ (дата звертання: 10.04.2022).
10. Omnibus GitLab documentation. [Електронний ресурс]. – Режим доступу: https://docs.gitlab.com/omnibus/ (дата звертання: 10.04.2022).
11. Enterprise Application Container Platform Docker[Електронний ресурс]. – Режим доступу: https://www.docker.com (дата звертання: 10.04.2022).
12. Docker Compose. [Електронний ресурс]. – Режим доступу: https://docs.docker.com/compose/ (дата звертання: 10.04.2022).
13. Зеркалов Д.В. Безпека життєдіяльності та основи охорони праці. Навчальний посібник. К.: «Основа». 2016. – 267 с.

14. Козлов С.С. Методичні вказівки до виконання розділу “Охорона праці та безпека в надзвичайних ситуаціях” в дипломних проектах для підготовки студентів факультету електроніки за освітньо-кваліфікаційним рівнем “Спеціаліст” та ”Магістр”. "Вимоги безпеки під час експлуатації обчислювальної техніки" / К.:НТУУ ”КПІ”, 2015, - 30 с.

15. Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями. [Електронний ресурс] - Режим доступу: https://zakon.rada.gov.ua/laws/show/z0508-18 (Дата звернення: 01.06.2022).