МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

«СИСТЕМИ КОНТРОЛЮ ВЕРСІЙ»

Лабораторна робота №1

“Управління ІТ-проектами”

для студентів базового напрямку 6.050101 “Комп’ютерні науки”

Студент: Сущенко Д. Ю.

Група: КН-410

Варіант: 22

Кафедра: САПР

Перевірила: Климкович Т. А.

Львів – 2022

Мета роботи:

Ознайомитись з системами контролю версій. Зрозуміти принципи синхронізації робочих груп у гнучких командах. Набути навичок використання технічних засобів та протоколів для роботи з системами контролю версій.

Завдання:

1. Ознайомитись з принципами роботи систем контролю версій.
2. Отримати індивідуальне завдання у викладача.
3. Написати програму згідно з індивідуальним завданням.
4. Створити репозиторій та завантажити туди свій програмний код.
5. Внести зміни в програму, при необхідності провести вирішення конфліктів.

Індивідуальне завдання:

Створення структури даних на основі LinkedList.

Відповіді на контрольні запитання:

1. **Що таке системи контролю версій? Для чого вони використовуються?**

   Система контролю версій – це програма, що надає утиліти для зручної організації, зберіганні, маніпуляції та перевірки змін файлів та папок. Використовується для організації роботи з змінами файлів та папок, а також, для коректності файлів на різних приладах.

2. **Що виконують команди checkout та clone?**

   Команда checkout – дозволяє змінити поточну гілку роботи проекту в локальному репозиторії.

   Команда clone – копіює гілку з віддаленого серверу, або локального комп’ютера у поточну директорію виклику команди.

3. **Що виконує команда commit?**

Команда commit – фіксує зміни поточного проекту.

4. **Що таке branch? Для чого вони використовуються?**

   За допомогою команди branch можна відбрунькуватись від основного стовбуру. Завдяки цій можливості усі подальші зміни можна зберігати та опрацьовувати окремо від стовбура не «ризикуючи» ним.

5. **Принципи виконання merge?**

   Команда merge дозволяє виконувати злиття двох гілок, де одна є «донором» файлі та папок, а інший «акцептор», що приймає усі виконані зміни.

6. **Що таке блокування? Які його недоліки?**

   Блокування, це можливість змінити статус файлу, який не дозволяє його змінювати інших розробникам окрім того, хто його заблокував. Через можливість блокування усім іншим розробникам доводиться чекати, поки закінчяться роботи над цим файлом, що призводить до затримки робочого процесу.

7. **Що таке tag? Для чого він використовується?**

   Tag – це символічна мітка, що ідентифікує поточну версію вітки за певними правилами розробників. В загальному, теги використовуються для ідентифікацію версії проекту, відношення до чогось.

8. **Який цикл розробки проекту з використанням VCS?**

   Будь-які робочі, тестові або демонстраційні версії проекту збираються тільки з репозиторію системи. Поточна версія головної гілки завжди коректна. Не допускається фіксація в головній гілці неповних або не пройшовши хоча б попереднє тестування змін. Будь-яка значима зміна має оформлятися як окрема гілка. Версії проекту позначаються тегами. Виділена і позначена тегом версія більш ніколи не змінюється.

9. **Які протоколи використовує Git?**

   Git використовує протоколи SSH, TCP та HTTP.

10. Що робить команда git –diff

Команда git -diff демонструє зміни між фіксованою версією та ще не зафіксованою.

11. Чим відрізняється команда git reset –-soft від git reset –-hard?

Команда git reset –soft дозволяє змінити версію проекту на певну зафіксовану версію без втрат проміжних версій між ними. Команда git reset –hard дозволяє те саме що і git reset –soft, але зафіксовані версії будуть видалені.

Хід роботи:

Для початку роботи на проектом був проініціалізований локальний репозиторій за допомогою команди ***git init*** (рис. 1).

```
C:\My\Projects\cpp\linkedlist>git init
Initialized empty Git repository in C:/My/Projects/cpp/linkedlist/.git/
```

Рис. 1 Ініціалізація репозиторію

Після чого було додані перші файли у проект (рис. 2) та додані у індекс git-а за допомогою команди ***git add –all*** (рис. 3) та зафіксовані за допомогою команди ***git commit -m “”*** (рис. 4).

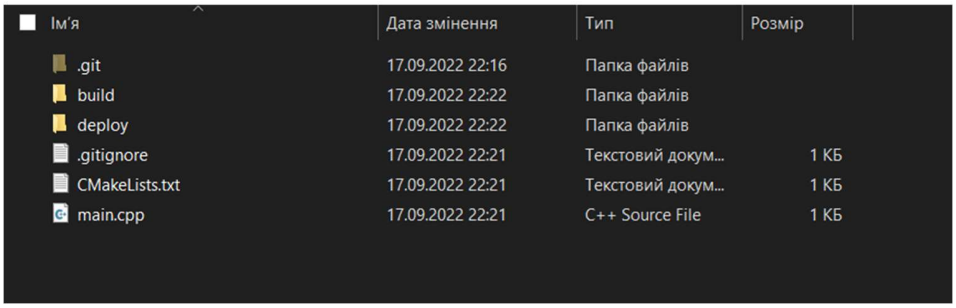

Рис. 2 Перші файли проекту

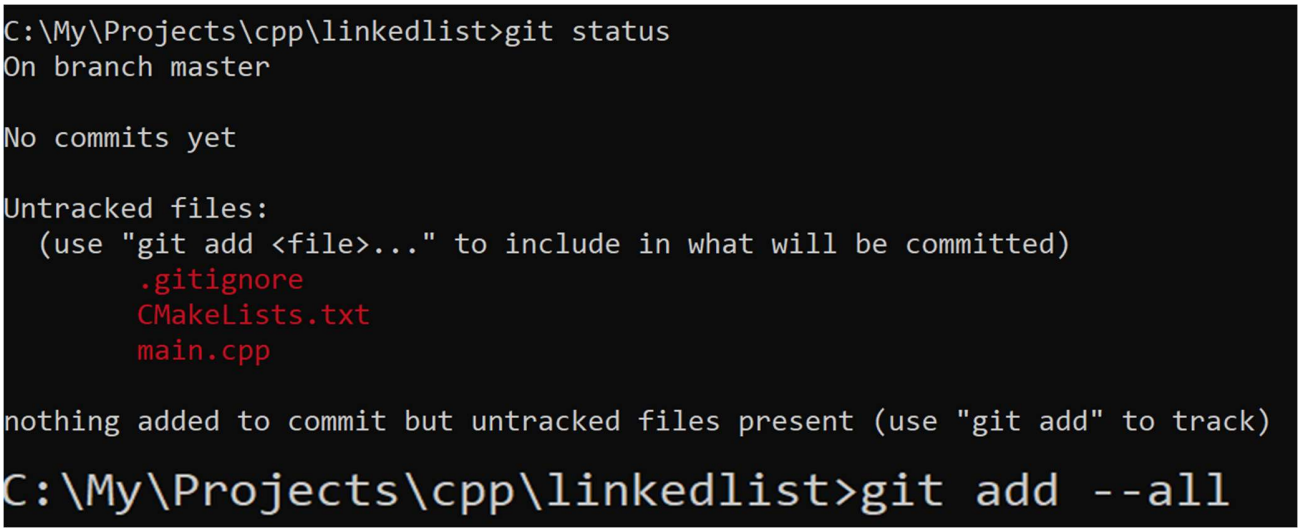

Рис. 3 Додані файли у індекс файлів

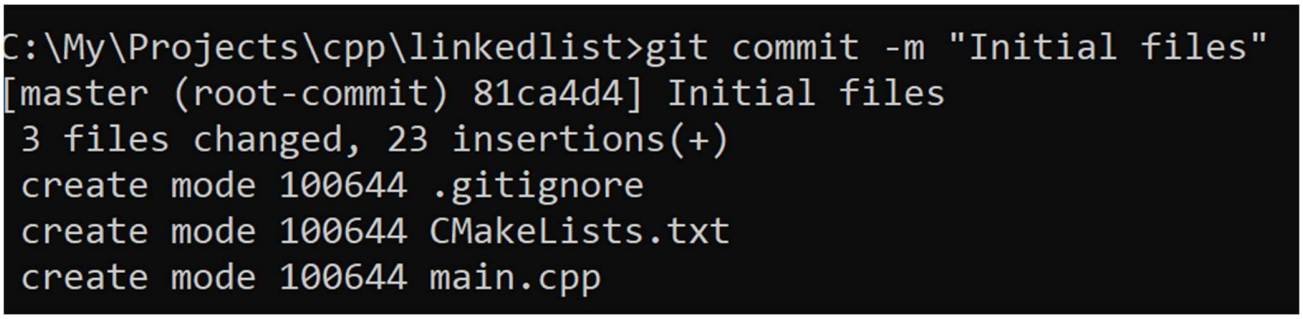

Рис. 4 Зафіксовані зміни

Коли було додані перші файли та зафіксовані зміни, треба було зберегти репозиторій на віддаленому сервері. В поточному проекті було обрано використовувати GitHub. За допомогою команди ***git push*** (рис. 5) стало зрозуміло, що треба спочатку створити майбутній віддалений репозиторій на сервері (рис. 6) і вже тільки тоді виконувати завантаження локального репозиторію (рис. 7).

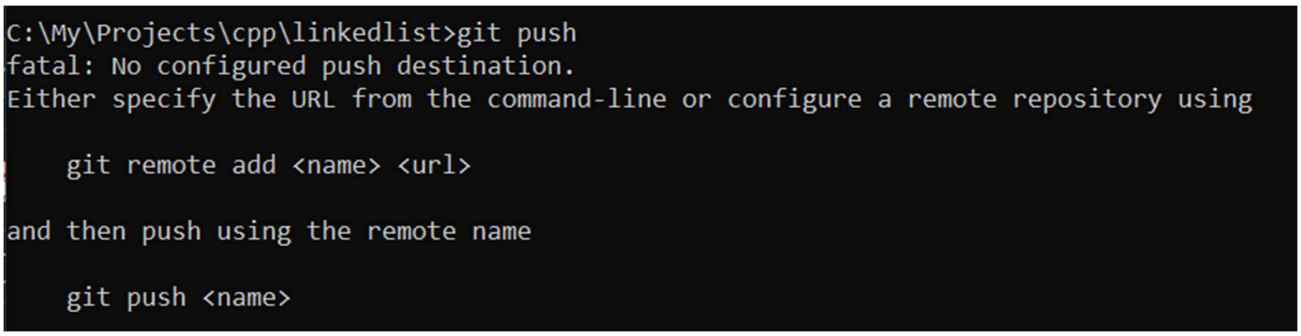

Рис. 5 Спроба завантажити лок. репозиторій на сервер

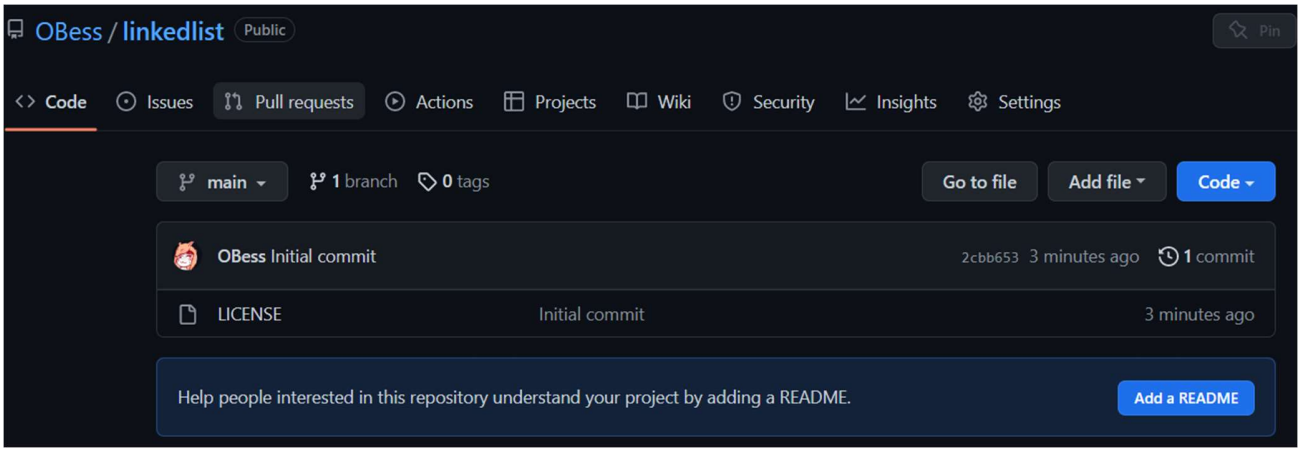

Рис. 6 Створений репозиторій на віддаленому сервері

```
C:\My\Projects\cpp\linkedlist>git remote add linkedlist https://github.com/OBess/linkedlist.git

C:\My\Projects\cpp\linkedlist>git push --set-upstream linkedlist master
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 8 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (8/8), 1.05 KiB | 1.05 MiB/s, done.
Total 8 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), done.
remote:
remote: Create a pull request for 'master' on GitHub by visiting:
remote:      https://github.com/OBess/linkedlist/pull/new/master
remote:
To https://github.com/OBess/linkedlist.git
 * [new branch]      master -> master
branch 'master' set up to track 'linkedlist/master'.
```

Рис. 7 Завантаження лок. репозиторію на сервер

Але виявилось, що створилась друга вітка (рис. 8), яка не потрібна на даний момент тому було вирішено провести злиття. Дане злиття було виконано не за допомогою команди ***git merge***, так як треба було створити історію не існуючою локальної вітки, тому було виконано команду ***git pull linked list –allow-unrelated-histories***(рис. 9).

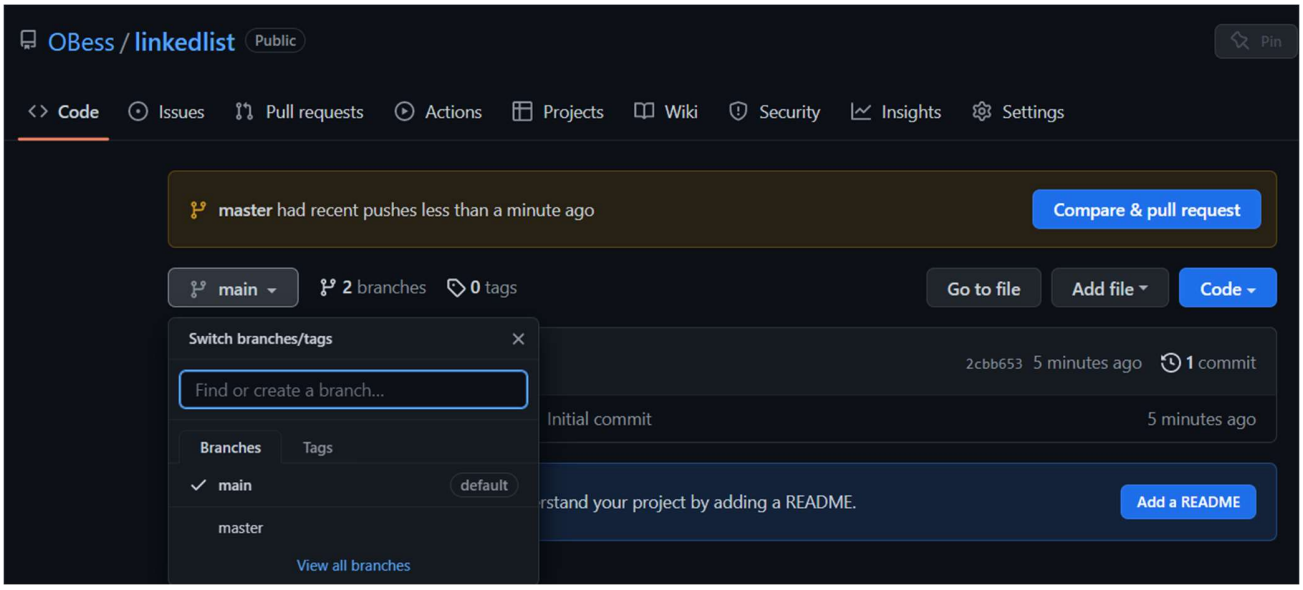

Рис. 8 Друга, не потрібна, вітка

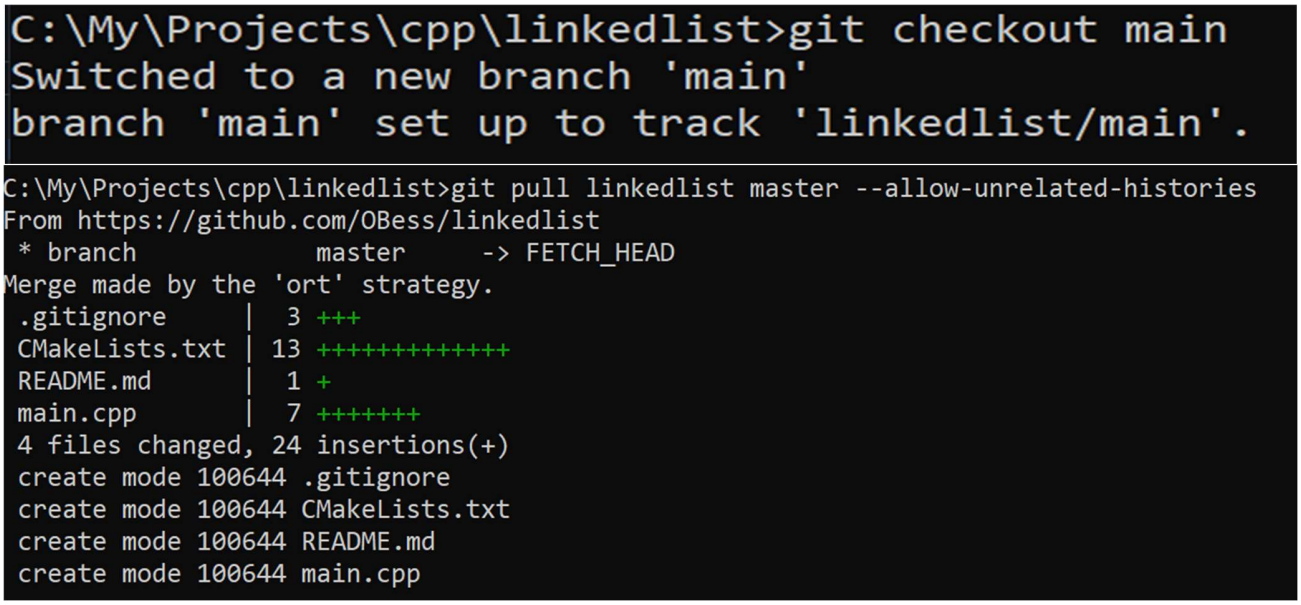

Рис. 9 Злиття віток

В результаті було отримано репозиторій на сервері з певними файлами (рис. 10), що тепер можна модифікувати в локальному репозиторію та завантажувати зміни на сервер.

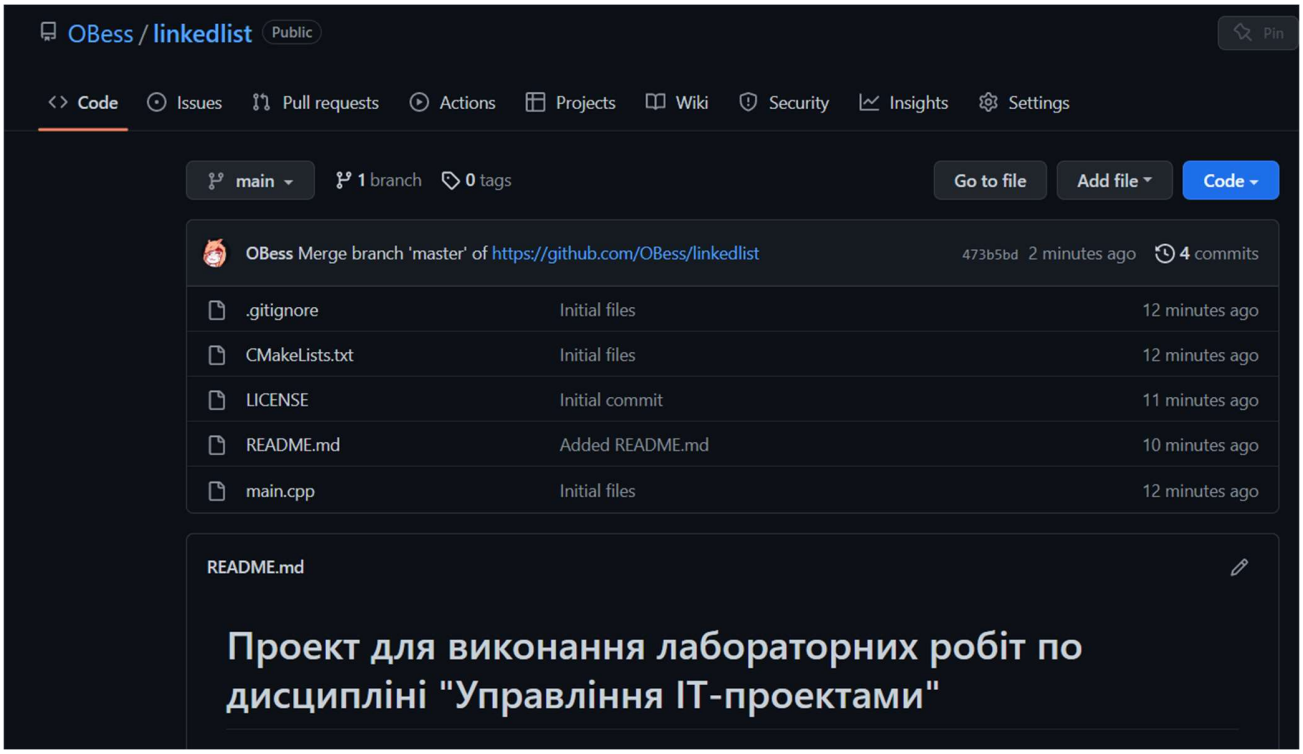

Рис. 10 Завантажений проект

Для оновлення проекту було додано файл include/linkedlist.hpp та модифіковано файл main.cpp (рис. 11), після чого всі ці файли додані (рис. 12), зафіксовані (рис. 13) та завантажені на сервер (рис. 14) і в результаті в нас оновлений проект, що продемонстровано на рис. 15.

```
C:\My\Projects\cpp\linkedlist>git status
On branch main
Your branch is up to date with 'linkedlist/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   .gitignore
        modified:   main.cpp

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        include/
        main.exe

no changes added to commit (use "git add" and/or "git commit -a")
```

Рис. 11 Оновлений проект

```
C:\My\Projects\cpp\linkedlist>git add --all
```

Рис. 12 Додані файли в індекс файлів

```
C:\My\Projects\cpp\linkedlist>git commit -m "Added LinkeList"
[main c8a6a08] Added LinkeList
 4 files changed, 304 insertions(+), 2 deletions(-)
 create mode 100644 include/linkedlist.hpp
```

Рис. 13 Зафіксовані поточні зміни

```
C:\My\Projects\cpp\linkedlist>git push
Enumerating objects: 12, done.
Counting objects: 100% (12/12), done.
Delta compression using up to 8 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (9/9), 55.78 KiB | 7.97 MiB/s, done.
Total 9 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/OBess/linkedlist.git
   473b5bd..20dc8c6  main -> main
```

Рис. 14 Завантажено зміни на сервер

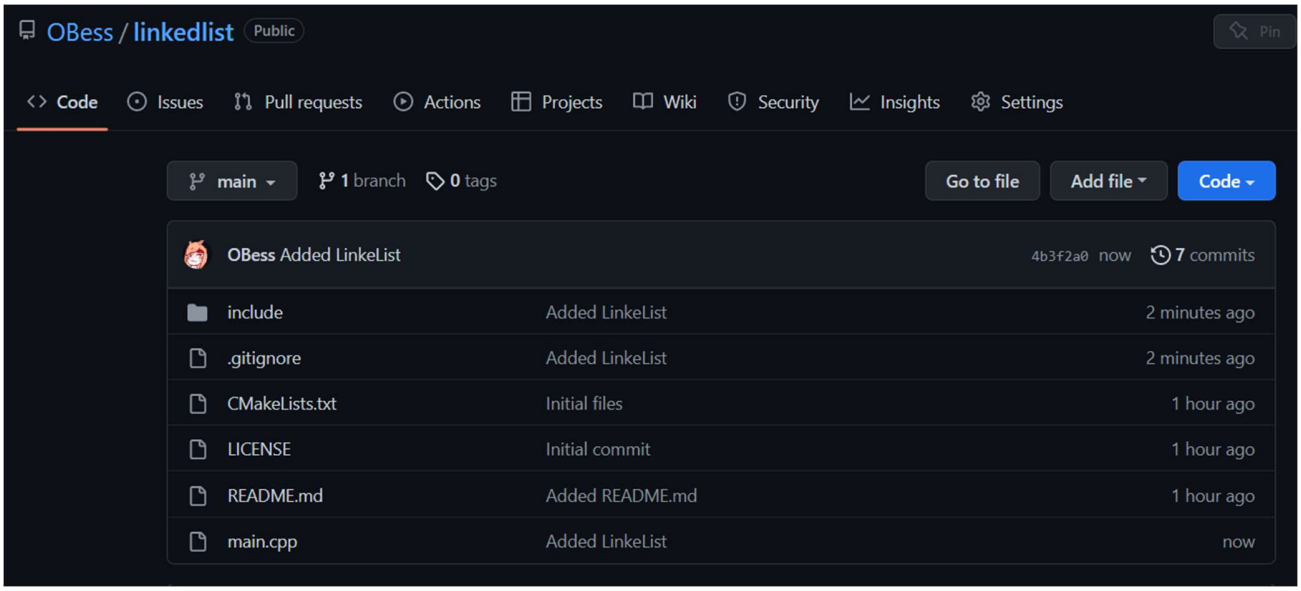

Рис. 15 Завантажені зміни на сервері

Висновок:

В ході роботи було створено проект на основі структури даних LinkeList за допомого системи контролю версій Git, де було збережено історію певних файлів та залиті на віддалений сервер для подальшого його розширення серед розробників. Репозиторій знаходиться за посиланням - https://github.com/OBess/linkedlist