Виконали студенти групи 2153ст:

Мамонова Зоряна

Фанькіна Алла

Тимошин Георгій

Білий Владислав

Комаровський Іван

Гриненко Олег

Мета: навчитися колективно використовувати систему керування версіями, розподілити відповідальність за виконання робіт між учасниками команди проекту.

Завдання: Вибрати систему для свого проекту, обґрунтувати свій вибір, показати хід роботи комадою проекту з вибраною СКВ.

Теоретичні відомості

Система керування версіями (СКВ) - це система, яка зберігає зміни в одному або декількох файлах так, щоб потім можна було відновити певні старі версії. Можна керувати версіями практично будь-яких типів файлів. Система керування версіями дозволяє повернути файли до колишнього вигляду, повернути до колишнього стану весь проект, порівняти зміни з якогось часу, побачити, хто останнім змінював модуль, який дав збій, хто створив проблему, і так далі. взагалі, якщо, користуючись СКВ, ви все зіпсували або втратили файли, все можна легко відновити.

Розподілені системи контролю версій

В таких системах як Git, Mercurial, Bazaar або Darcs клієнти не просто забирають останні версії файлів, а повністю копіюють репозиторій. Тому в разі, коли «вмирає» сервер, через який йшла робота, будь клієнтський репозиторій може бути скопійований назад на сервер, щоб відновити базу даних. Щоразу, коли клієнт забирає свіжу версію файлів, створюється повна копія всіх даних.

Крім того, в більшій частині цих систем можна працювати з декількома віддаленими репозиторіями, таким чином, можна одночасно працювати по-різному з різними групами людей в рамках одного проекту. Так, в одному проекті можна одночасно вести кілька типів робочих процесів, що неможливо в централізованих системах.

Git та Subversion В даний час більшість проектів з відкритим вихідним кодом, а також велика кількість корпоративних проектів, використовують Subversion для управління своїм вихідним кодом. Це найпопулярніша на поточний момент система управління версіями з відкритим вихідним кодом. Крім того, вона дуже схожа на CVS, систему, яка була найпопулярнішою до Subversion. Одна з чудових особливостей Git - можливість двостороннього обміну з Subversion через інтерфейс, званий Git SVN. Цей інструмент дозволяє вам використовувати Git в якості коректного клієнта при роботі з сервером Subversion. Таким чином, ви можете користуватися всіма локальними можливостями Git, а потім зберігати зміни на сервері Subversion так, як якщо б використовували Subversion локально. Тобто ви можете робити локальне розгалуження і злиття, використовувати індекс, переміщення і відбір патчів для переносу з однієї гілки в іншу (зняття вершків) і т.д., в той час, як ваші колеги будуть продовжувати використовувати в розробці підхід часів кам'яного віку. Це хороший спосіб протягти Git в робоче оточення своєї компанії, щоб допомогти колегам розробникам стати більш ефективними, в той час як ви будете лобіювати перехід повністю на Git. Інтерфейс обміну з Subversion це ворота в світ розподілених систем керування версіями.

Хід роботи

1. У якості веб-сервісу для хостінгу було обрано github (github.com). Причин обрати саме його було декілька:

* сайт має елементи соціальної мережі — крім розміщення коду розробники можуть спілкуватися, коментувати правки один одного, слідкувати за новинами знайомих;
* існує можливість об'єднувати різні репозиторії, в тому числі і такі, що мають різних власників;
* для проектів є персональні сторінки, Вікі та система відслідковування помилок. Прямо на сайті можна переглядати файли проектів, реалізовано підсвічування синтаксису для більшості мов програмування;
* можна працювати з сервісом через SSL та показувати репозиторій тільки визначеним користувачам. (На платних тарифних планах);
* сам код проектів можна отримати не тільки через Git, а й завантажити як звичайний архівний файл.

1. Основний репозиторій було відкрито користувачем zorgmeow, його назва – «Encryption».

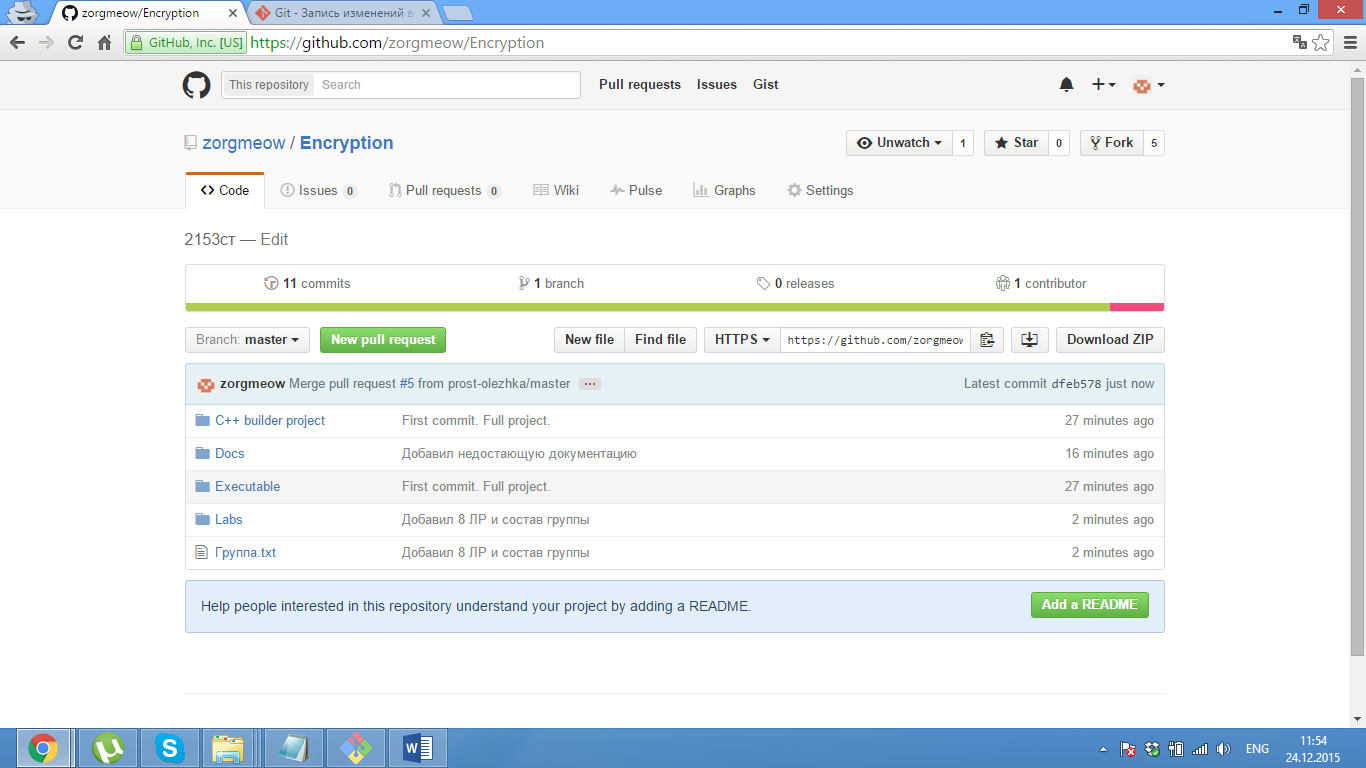


Рисунок 1 – Вигляд файлів проекту на github

1. Завантажили першу версію проекту одразу після створення репозиторію. Для завантаження використовувався Git Shell – консольна программа для користування Git.

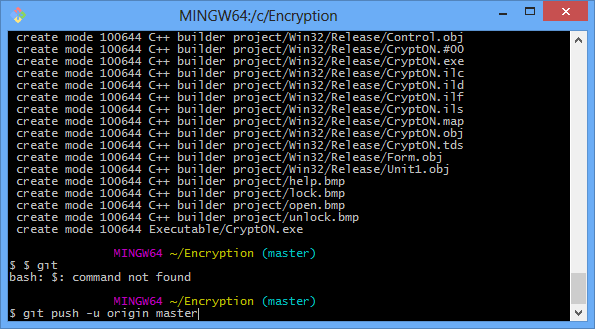


Рисунок 2 – Інтерфейс Git Shell

1. Основну гілку форкнули всі учасники команди, внесли зміни до деяких файлів, а потім надіслали запит на злиття гілок (merge).

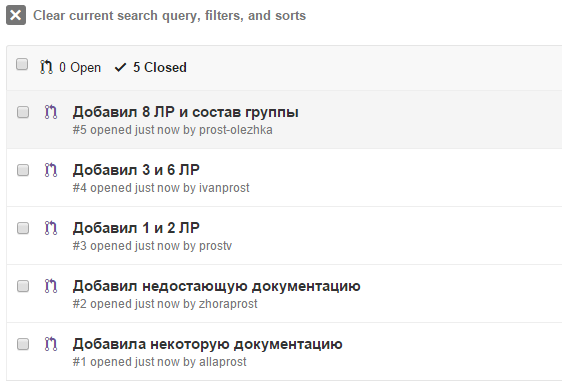


Рисунок 3 – Прийняті запити на злиття

Якщо користувач буде пропонувати злиття гілок, github сповістить про це.

Висновок: удосконалили навички з користування системами контролю версій. Git – найнадійніша та найзручніша система контролю версій на сьогоднішній день. Розробники Git приклали достатньо зусиль для того, щоб забезпечити розробників необхідною документацію для початку роботи з Git та його продуктивного використання. Gihub – дуже зручний хостінг (звичайно, це більше за хостінг) для проектів. Він дозволяє переглядати та завантажувати проекти без використання Git, а також регулярно надсилає сповіщення про будь-які зміни в репозиторії.