

Министерство Просвещения Республики Молдова
Технический Университет Молдовы

ОТЧЕТ

по лабораторной работе nr. 1
по предмету PR по теме:
“Версификация исходного кода с помощью GIT”

Выполнил:

ст. гр. TI-145 Ялтыченко А.

Проверил:

преп. Остапенко С.

Кишинев 2017 г.

Цель работы: Изучение и понимание принципов функционирования и использования системы контроля версий (VCS), известной как GIT.

Тема работы: Создание и настройка удаленного репозитория.

Задание:

- Создание удаленного репозитория, локализованного на github и синхронизация всех изменений произведенных локально;
- Создание проекта Maven с добавлением зависимости и отправкой на github.

Краткая теория

Системы версификации служат для управления множественными версиями файлов, включенных в групповой проект. Каждое действие над компонентом проекта сохраняется вместе с именем автора изменений. Важно отметить, что в любой момент можно откатиться к предыдущему состоянию этого компонента.

Ключевая мотивация использовать GIT состоит в возможности параллельной работы над проектом множества членов команды, пусть даже разнесенных географически на тысячи километров друг от друга. Помимо этого, существуют и другие преимущества. Так, если обнаруживается баг, всегда можно откатиться к предыдущей версии, отследить развитие проекта по ветвям, разрабатываемым параллельно.

Существуют две модели VCS:

- Централизованная модель (например, SVN): исходный код расположен на единственном центральном сервере, откуда клиенты могут получить рабочие варианты на свои локальные ПК. После осуществления изменений, разработчик запрашивает актуализацию серверного варианта кода;
- Распределительный (например, GIT): отсутствует единый центральный сервер. Синхронизация осуществляется на уровне “peer-to-peer”.

Ход работы

Создание удаленного репозитория

Для создания удаленного репозитория была произведена авторизация на веб-сайте GitHub и посредством графического интерфейса выбрана соответствующая опция. После того, как был создан сам репозиторий, SSH-ключ ПК, с которого велась разработка, был добавлен в аккаунт, как показано ниже на рис. 1:

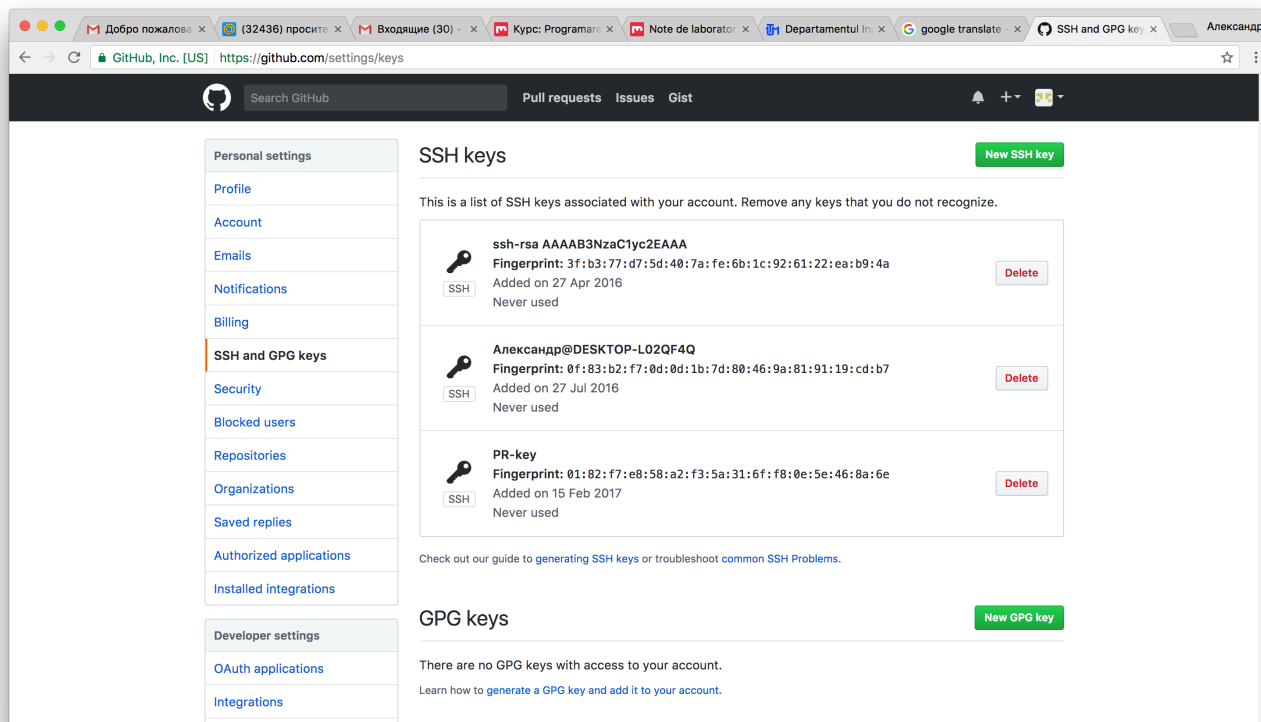


Рис. 1 – Добавление SSH-ключа

Инициализация локального репозитория

Следуя представленной на сайте GIT инструкции, локальный репозиторий был получен путем клонирования по ссылке командой вида:

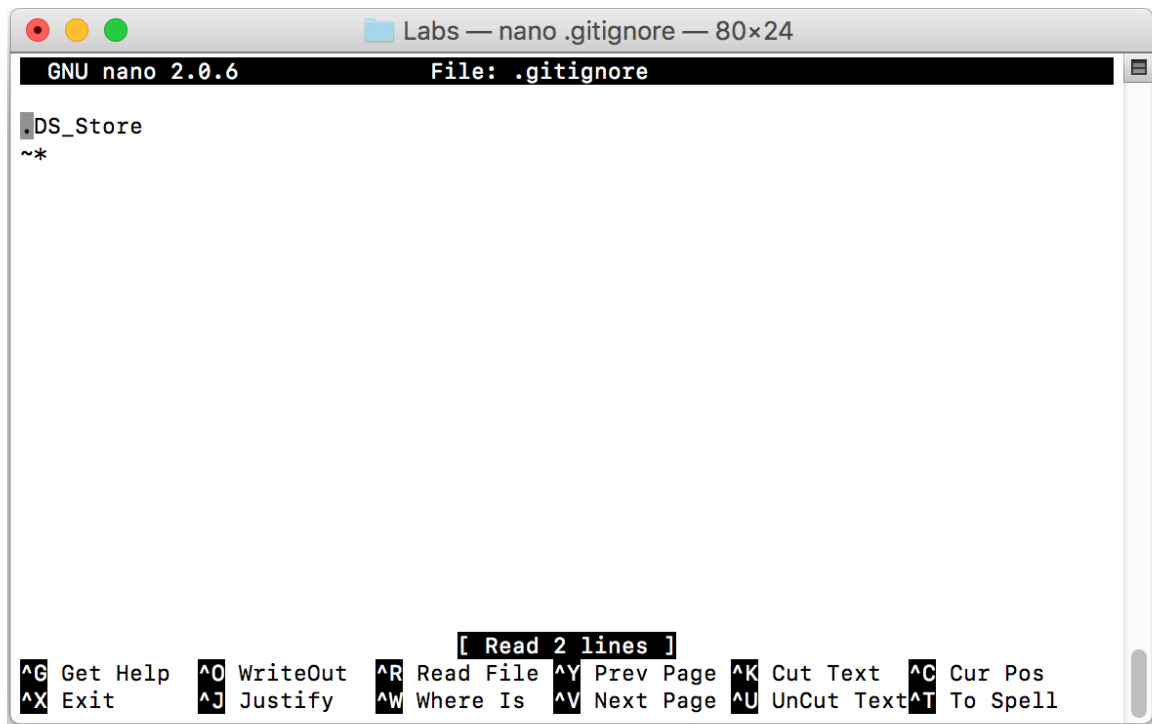
```
git clone https://github.com/AlexandrYaltychenko/PR.git
```

Затем были произведены первый commit и отправка на сервер:

```
git add .  
git commit -m "Initial commit"  
git remote add origin https://github.com/AlexandrYaltychenko/PR.git  
git push origin master
```

Создание .gitignore

Для того, чтобы избежать добавления в репозиторий системных и/или временных файлов, был создан и определен файл .gitignore, как показано ниже на рис. 2:



The screenshot shows a terminal window titled "Labs — nano .gitignore — 80x24". The editor is GNU nano 2.0.6. The file being edited is .gitignore. The content of the file is: `.DS_Store` followed by a new line and `~*`. The status bar at the bottom indicates "[Read 2 lines]" and lists various keyboard shortcuts for navigation and editing.

```
GNU nano 2.0.6 File: .gitignore

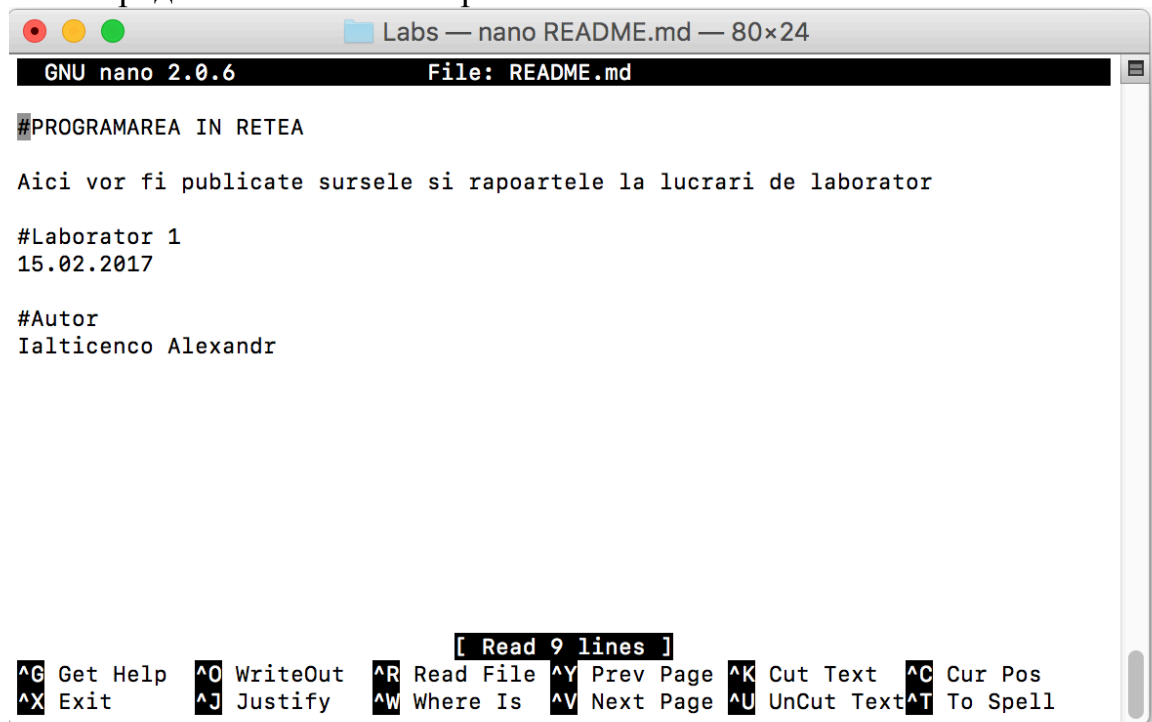
.DS_Store
~*

[ Read 2 lines ]
^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell
```

Рис. 2 – Создание .gitignore

Создание README.md

Для удобства ознакомления с репозиторием, а также документирования и структуризации хранящейся в нем информации был создан файл README.md. Его содержание представлено ниже на рис. 3:



The screenshot shows a terminal window titled "Labs — nano README.md — 80x24". The editor is GNU nano 2.0.6. The file being edited is README.md. The content of the file is: `#PROGRAMAREA IN RETEA`, a blank line, `Aici vor fi publicate sursele si rapoartele la lucrari de laborator`, a blank line, `#Laborator 1`, `15.02.2017`, a blank line, `#Autor`, and `Ialticenco Alexandr`. The status bar at the bottom indicates "[Read 9 lines]" and lists various keyboard shortcuts for navigation and editing.

```
GNU nano 2.0.6 File: README.md

#PROGRAMAREA IN RETEA

Aici vor fi publicate sursele si rapoartele la lucrari de laborator

#Laborator 1
15.02.2017

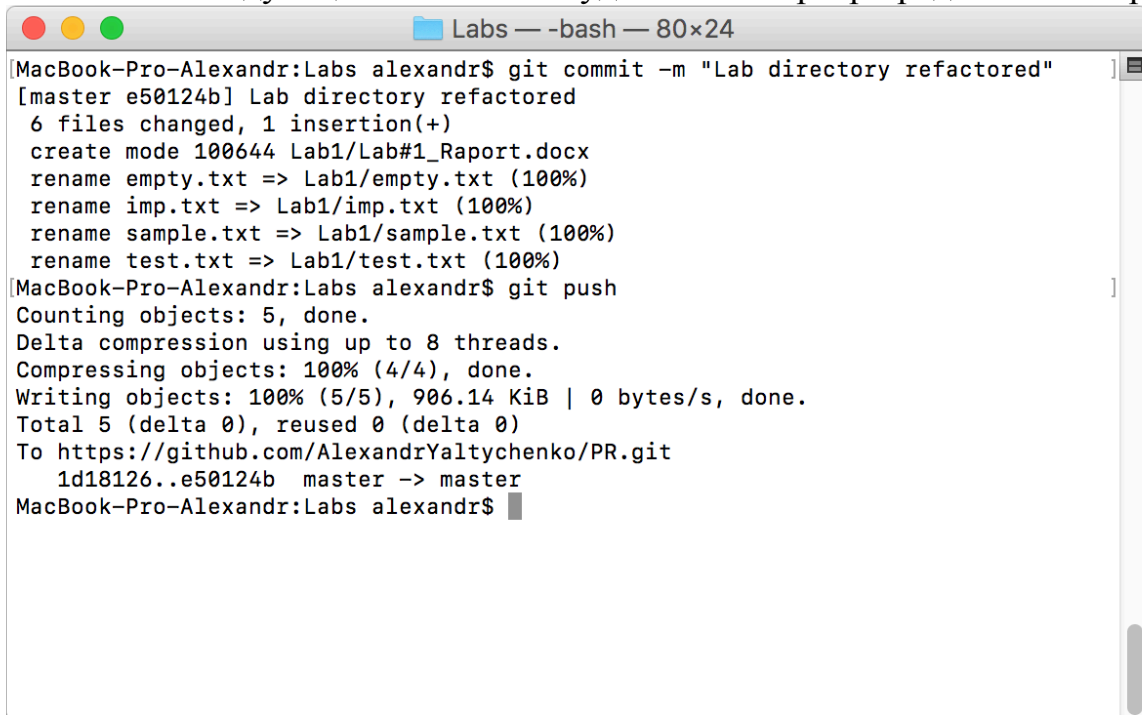
#Autor
Ialticenco Alexandr

[ Read 9 lines ]
^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell
```

Рис. 3 – Файл README.md

Работа с репозиторием

Далее производилась работа с репозиторием в нескольких ветках. Пример commit-а изменений с их последующей загрузкой на удаленный сервер представлен на рис. 4:



```
MacBook-Pro-Alexandr:~ alexandr$ cd Labs
MacBook-Pro-Alexandr:Labs alexandr$ git commit -m "Lab directory refactored"
[master e50124b] Lab directory refactored
6 files changed, 1 insertion(+)
create mode 100644 Lab1/Lab#1_Raport.docx
rename empty.txt => Lab1/empty.txt (100%)
rename imp.txt => Lab1/imp.txt (100%)
rename sample.txt => Lab1/sample.txt (100%)
rename test.txt => Lab1/test.txt (100%)
MacBook-Pro-Alexandr:Labs alexandr$ git push
Counting objects: 5, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (5/5), 906.14 KiB | 0 bytes/s, done.
Total 5 (delta 0), reused 0 (delta 0)
To https://github.com/AlexandrYaltychenko/PR.git
 1d18126..e50124b master -> master
MacBook-Pro-Alexandr:Labs alexandr$
```

Рис. 4 – Работа с репозиторием

Наконец, две ветви были слиты с помощью merge, таким образом, что был получен граф изменений, представленный ниже на рис. 5:

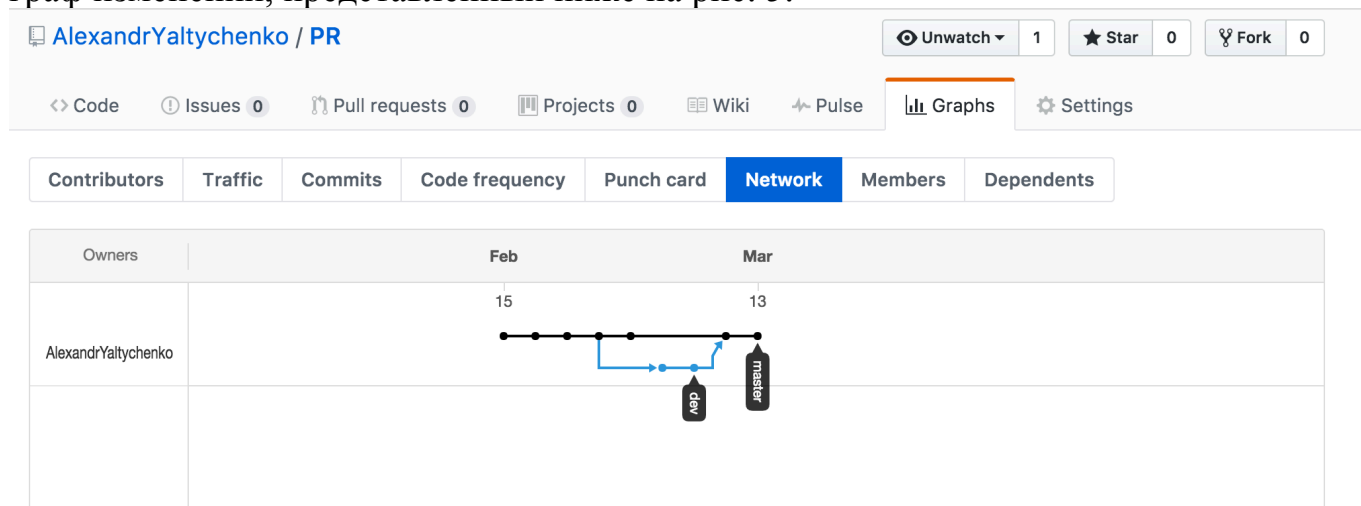


Рис. 5 – Граф изменений

Создание Maven-проекта

Для разработки Maven-проекта использовалась IDE IntelliJ. Создание нового проекта произведено по нажатию “Create New Project” в Welcome Dialog, как показано на рис. 6:

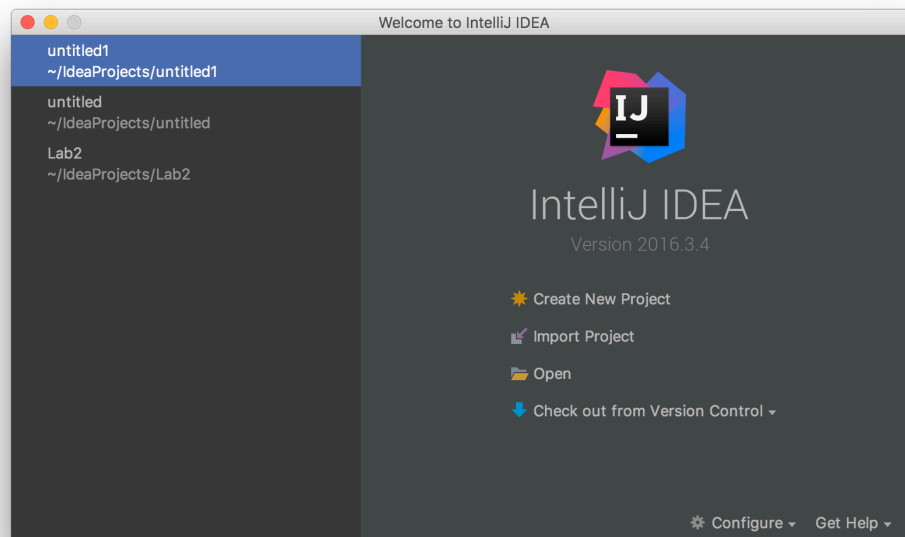


Рис. 6 – Создание проекта через Welcome Dialog

Написание и компиляция простейшего кода

Простейший код, отображающий приветствие написан в виде одного стандартного статического метода (`public static void main`) внутри `Main`-класса, как показано на рис. 7:

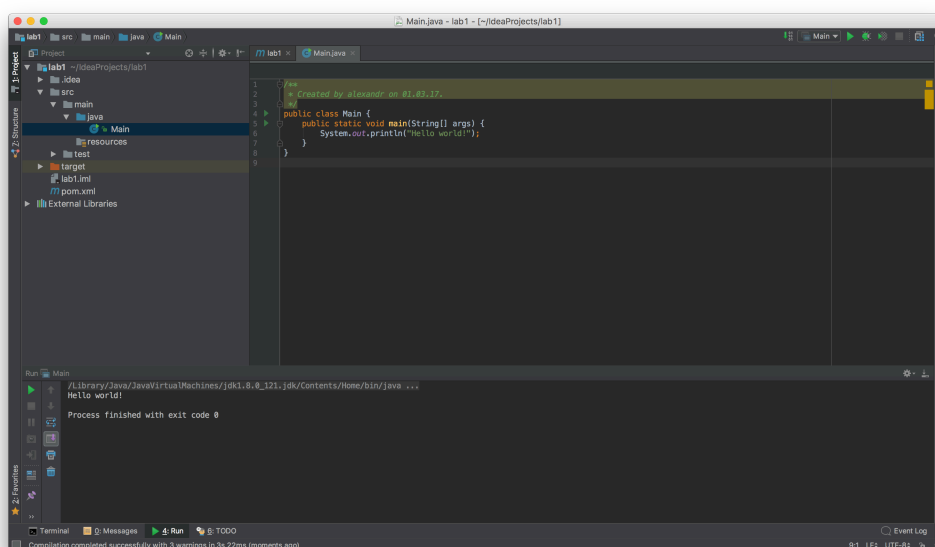


Рис. 7 – Код приветствия

Добавление зависимостей

В качестве зависимостей добавлены стандартно используемый почти любым java-проектом junit, а также удобнейшую библиотеку для работы с форматом json – `javaх.json`. Результат редактирования файла `pom.xml` представлен ниже на рис. 8:

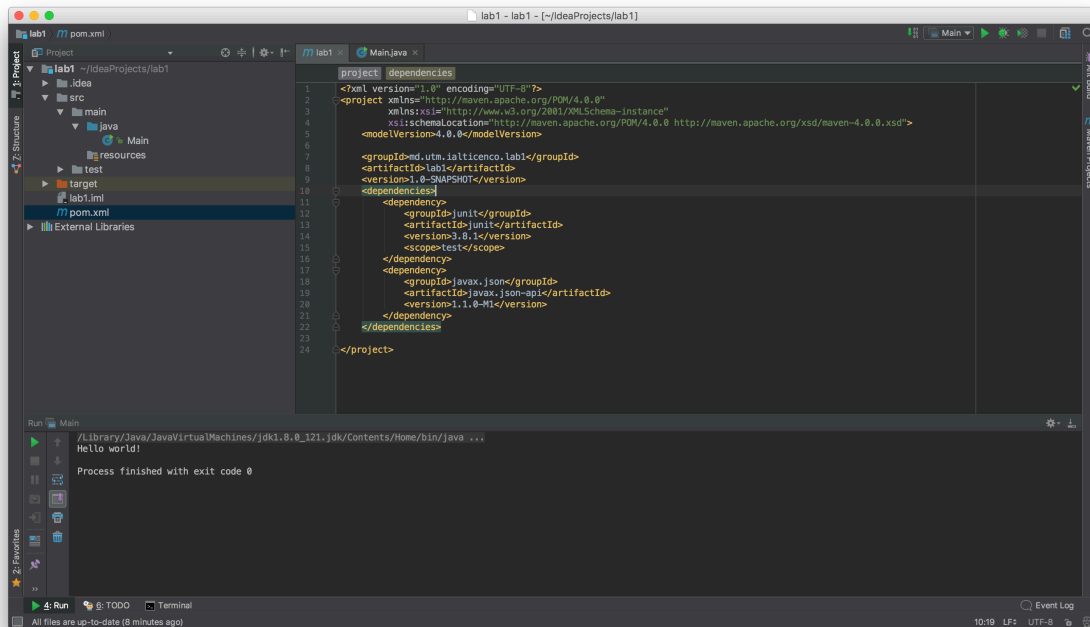


Рис. 8 – Измененный файл pom.xml

Добавление в GIT-репозиторий

Поскольку проект располагается по адресу являющемуся подпапкой существующего репозитория, сама IDE предложила настроить root для данного репозитория, как показано на рис. 9:

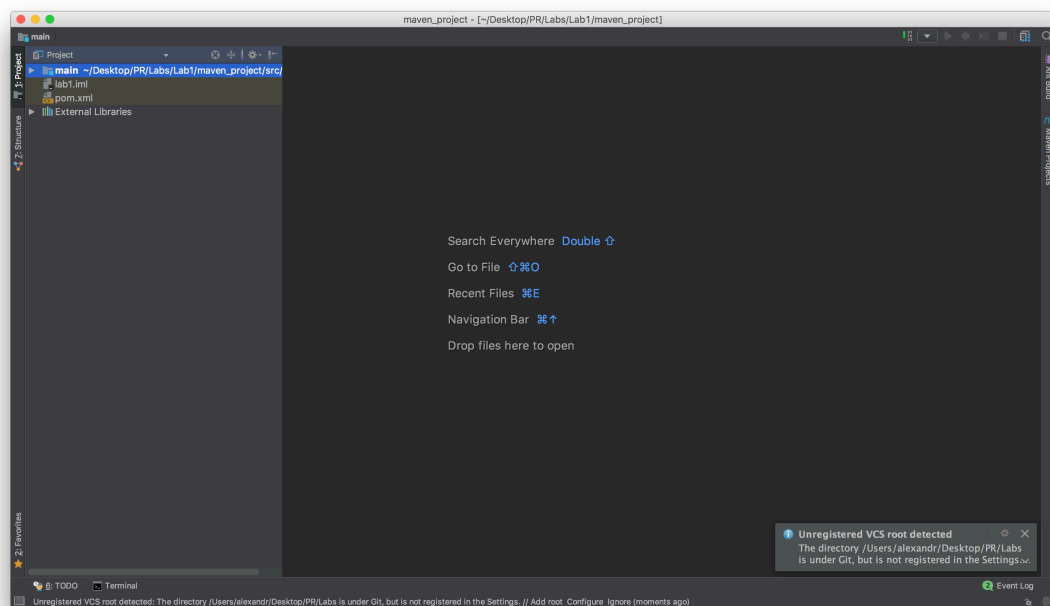


Рис. 9 – IntelliJ IDE обнаружение незарегистрированного VCS

Осталось лишь произвести коммит необходимых изменений, выбрав файлы, подлежащие версифицированию, как показано на рис. 10:

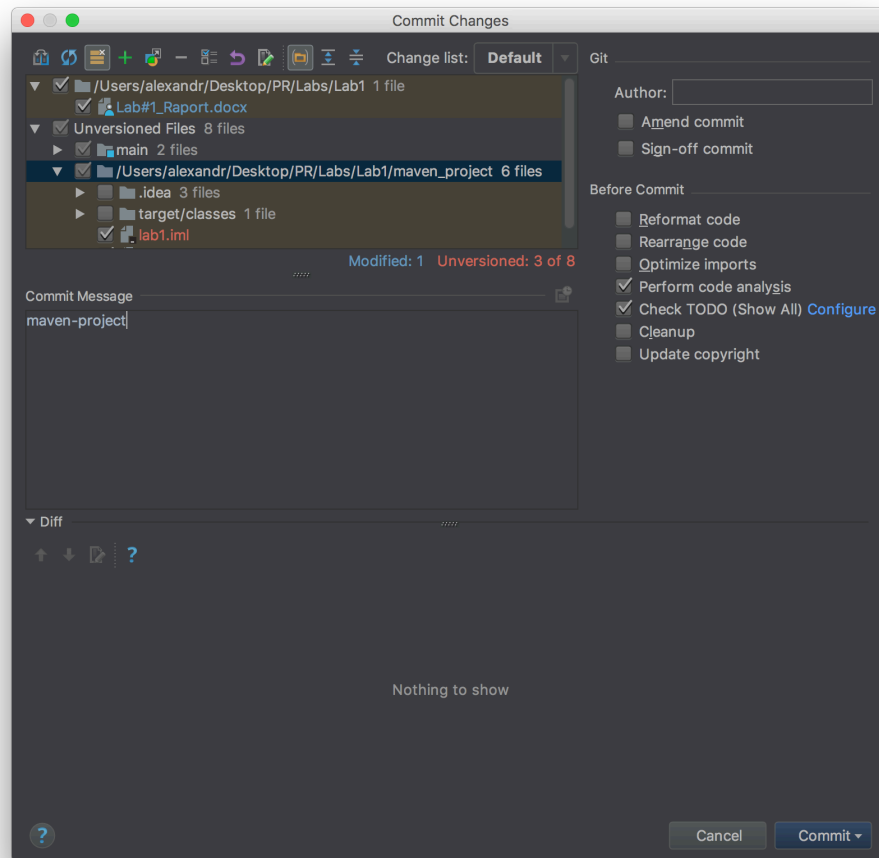


Рис. 10 – Осуществление GIT commit средствами IntelliJ IDE

Теперь осталось лишь отправить последние изменения на удаленный сервер с помощью соответствующего диалога (VCS -> GIT -> PUSH), как показано на рис. 11:

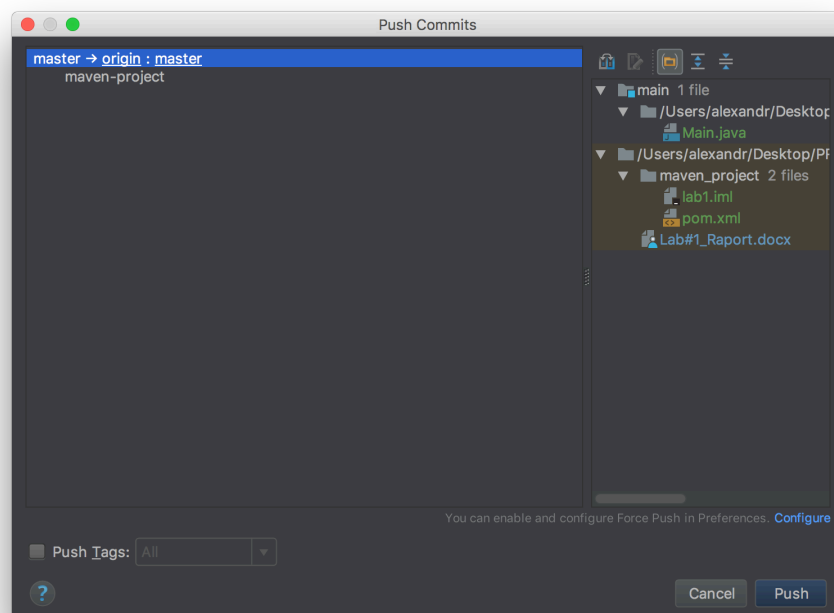


Рис. 11 – Отправка изменений на удаленный сервер через диалог IntelliJ IDEA

Выводы

В рамках данной лабораторной работы были изучены базовые принципы функционирования и использования системы версификации исходного кода GIT, а также системы сборки Maven. На базе полученных знаний был создан и настроен удаленный репозиторий с двумя ветвями, соответствующий условиям задания, а также реализован простейший Maven проект с подключенными к нему двумя сторонними зависимостями. Замечено, что использование системы версификации исходного кода заметно снижает риски утраты важной информации, оптимизирует и ускоряет процесс командной разработки, а система сборки в свою очередь позволяет избежать дополнительных временных и умственных затрат на ручную загрузку и подключение библиотек.