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, локализированного 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:

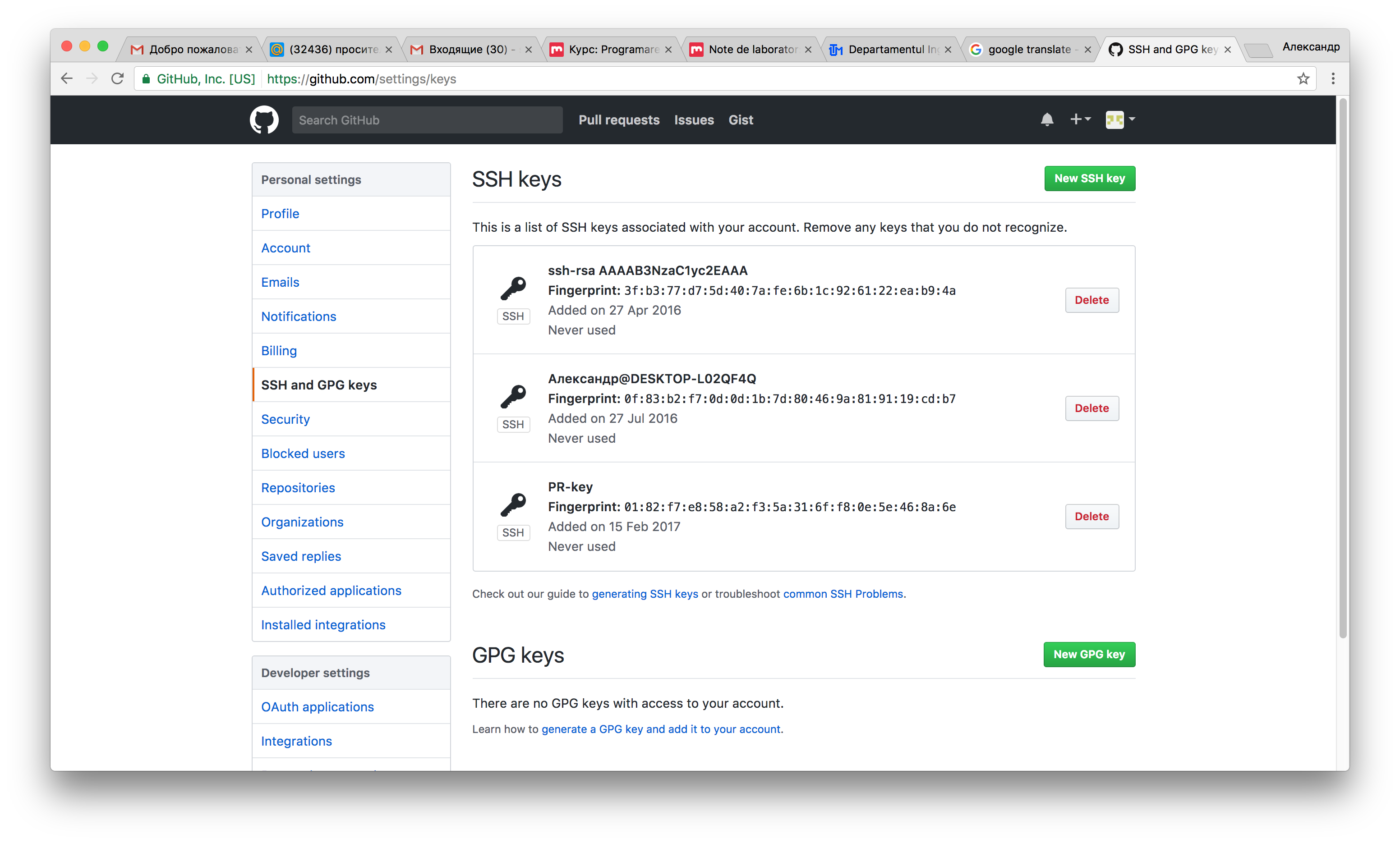


Рис. 1 – Добавление SSH-ключа

**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:



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:

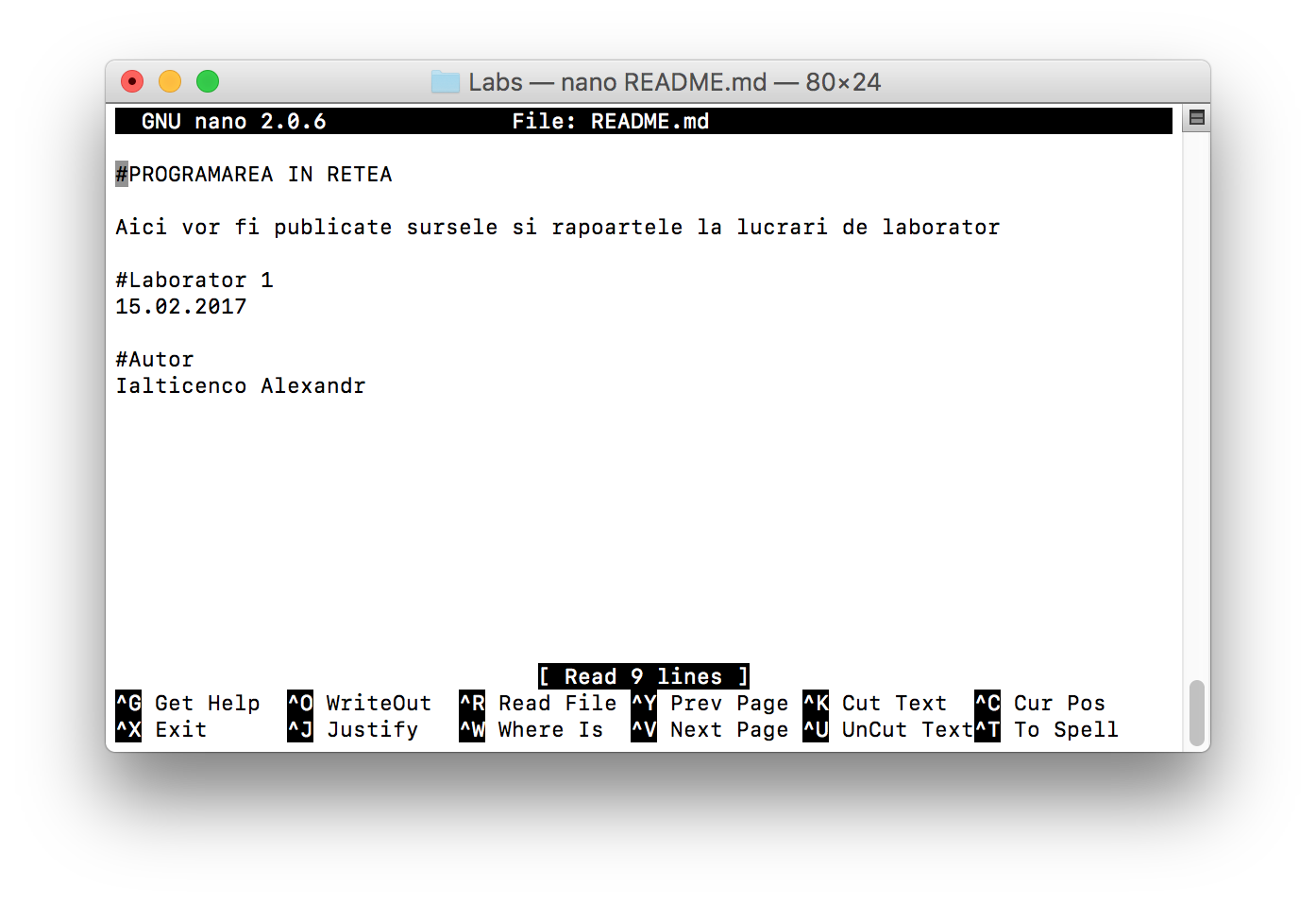


Рис. 3 – Файл README.md

**Работа с репозиторием**

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

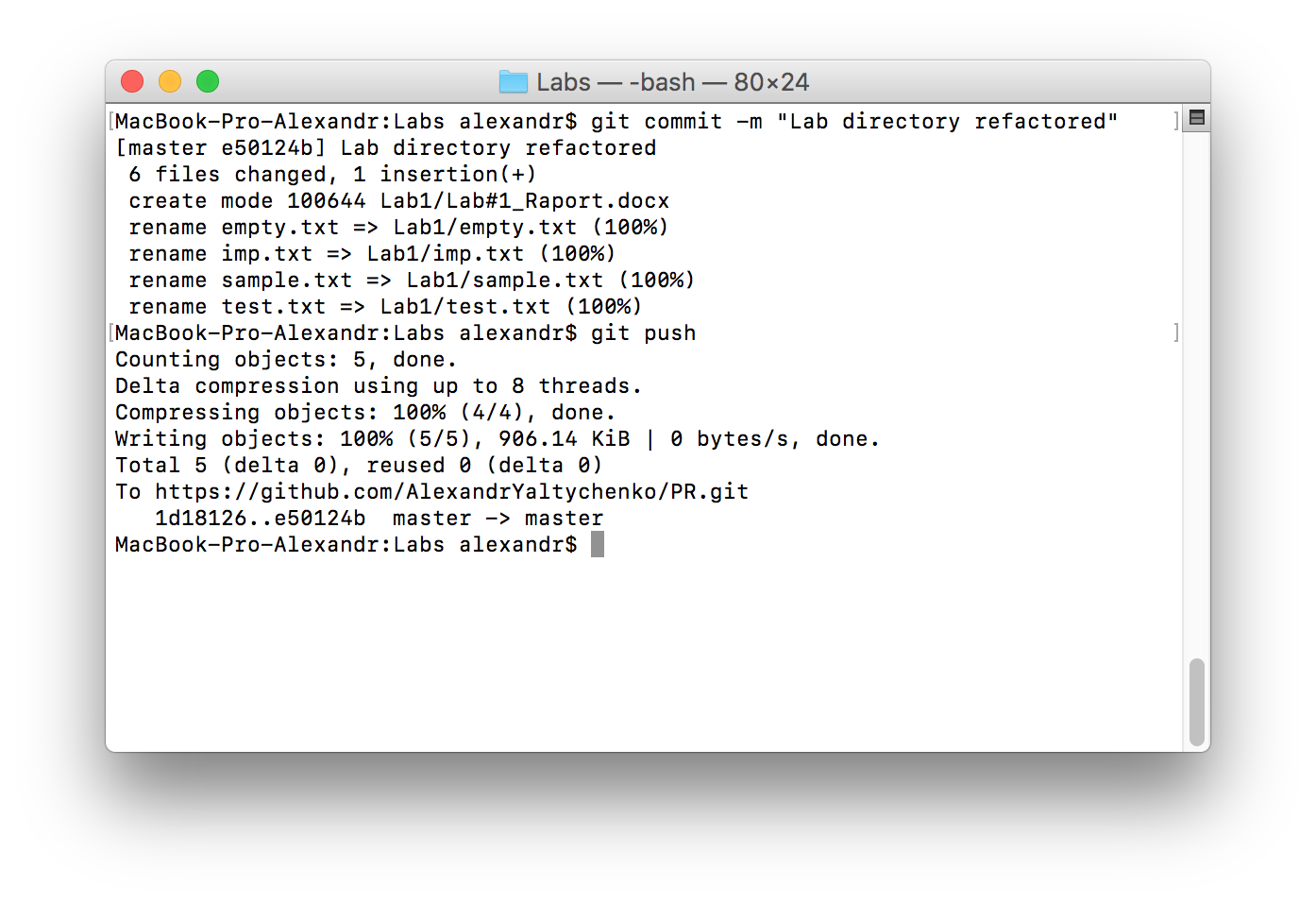


Рис. 4 – Работа с репозиторием

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

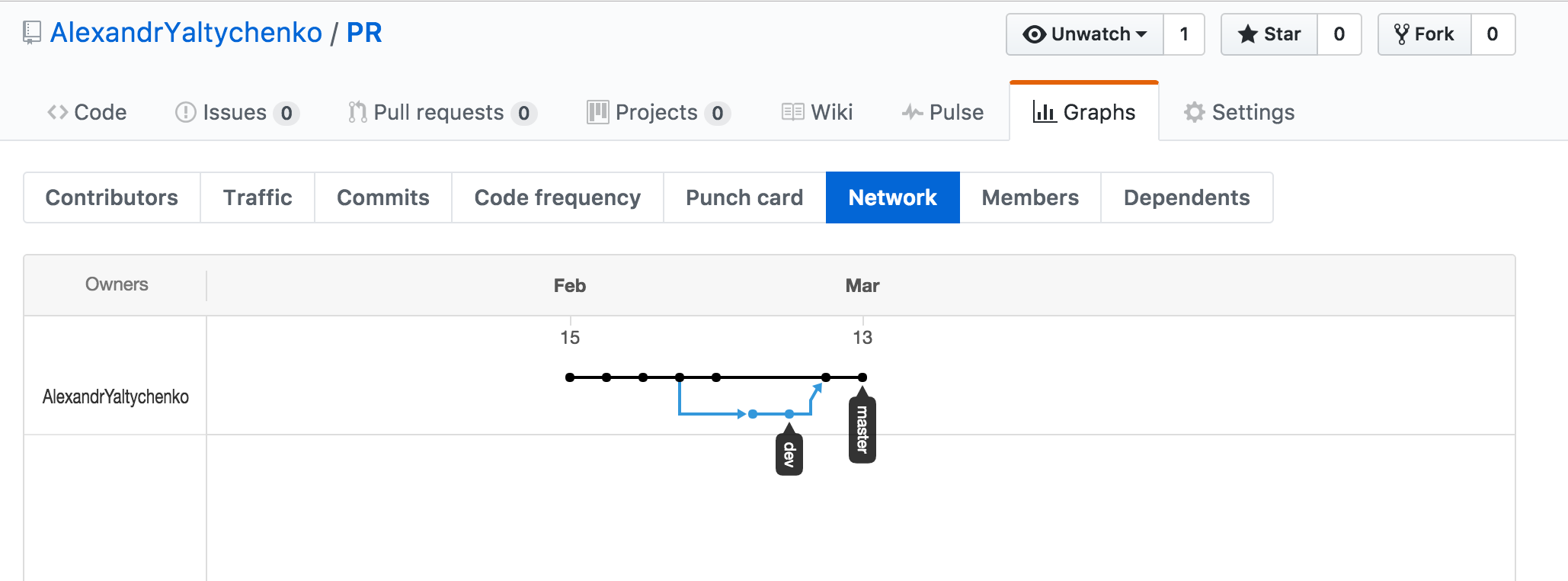


Рис. 5 – Граф изменений

**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:

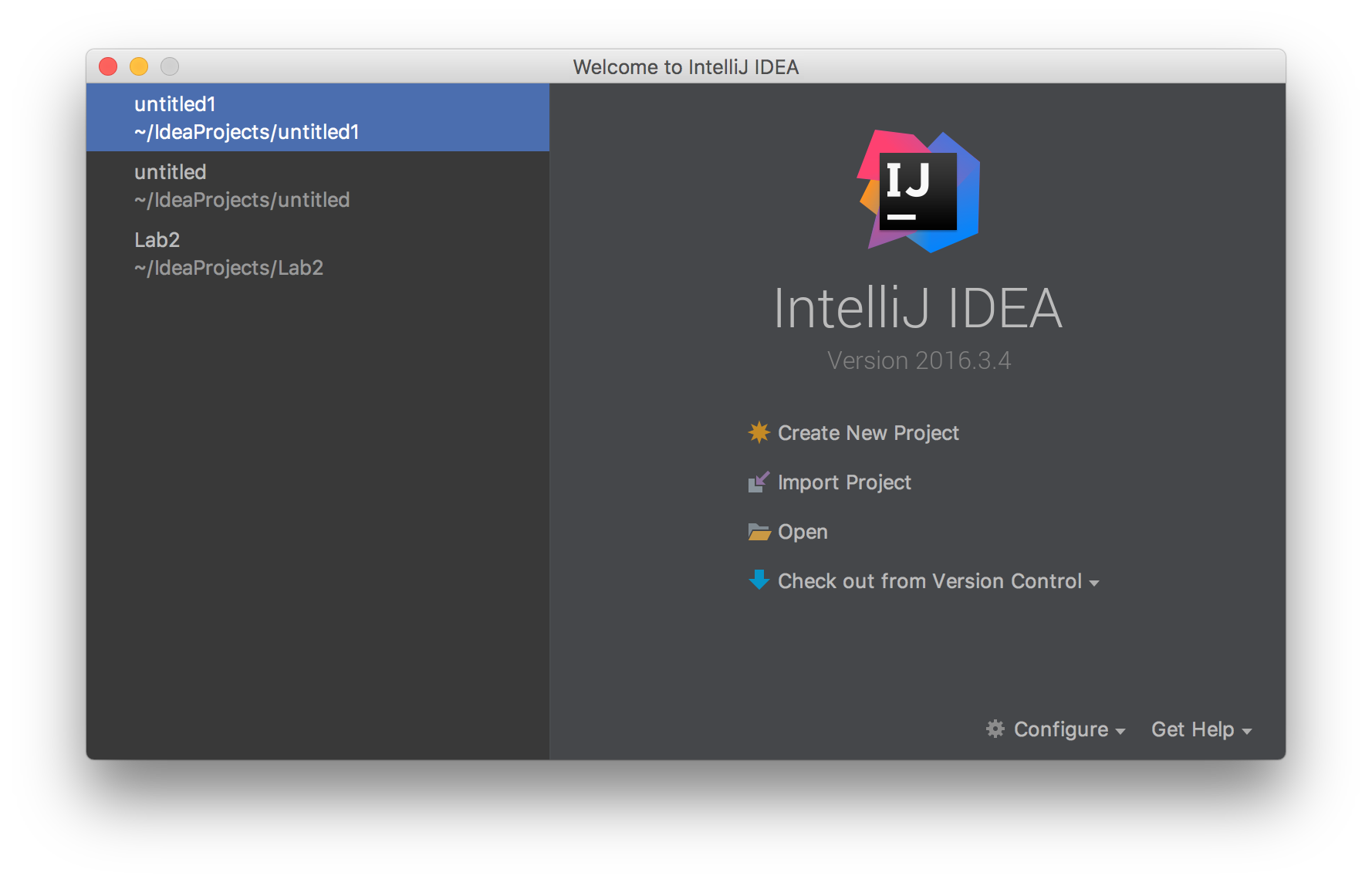


Рис. 6 – Создание проекта через Welcome Dialog

**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:

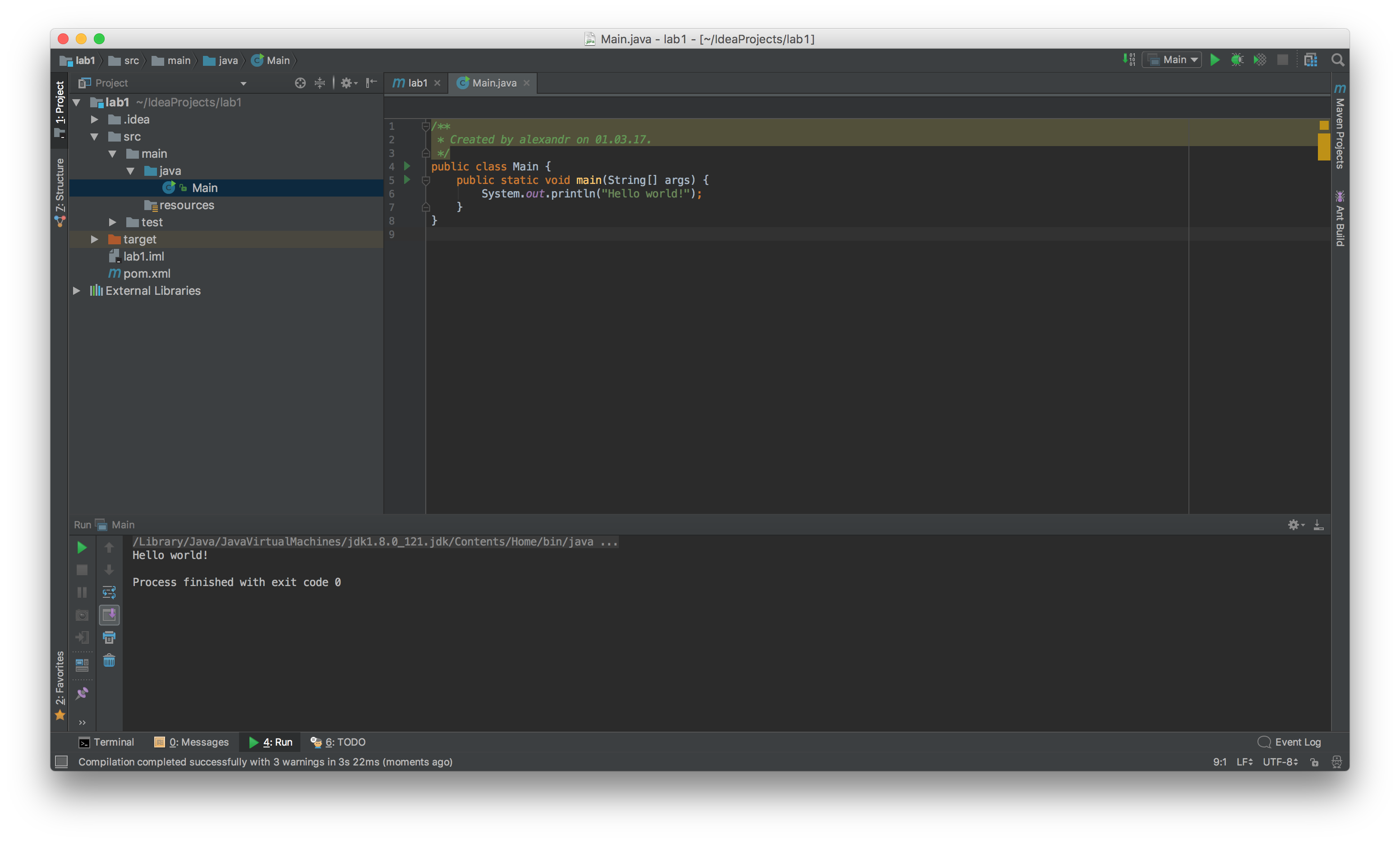


Рис. 7 – Код приветствия

**Добавление зависимостей**

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

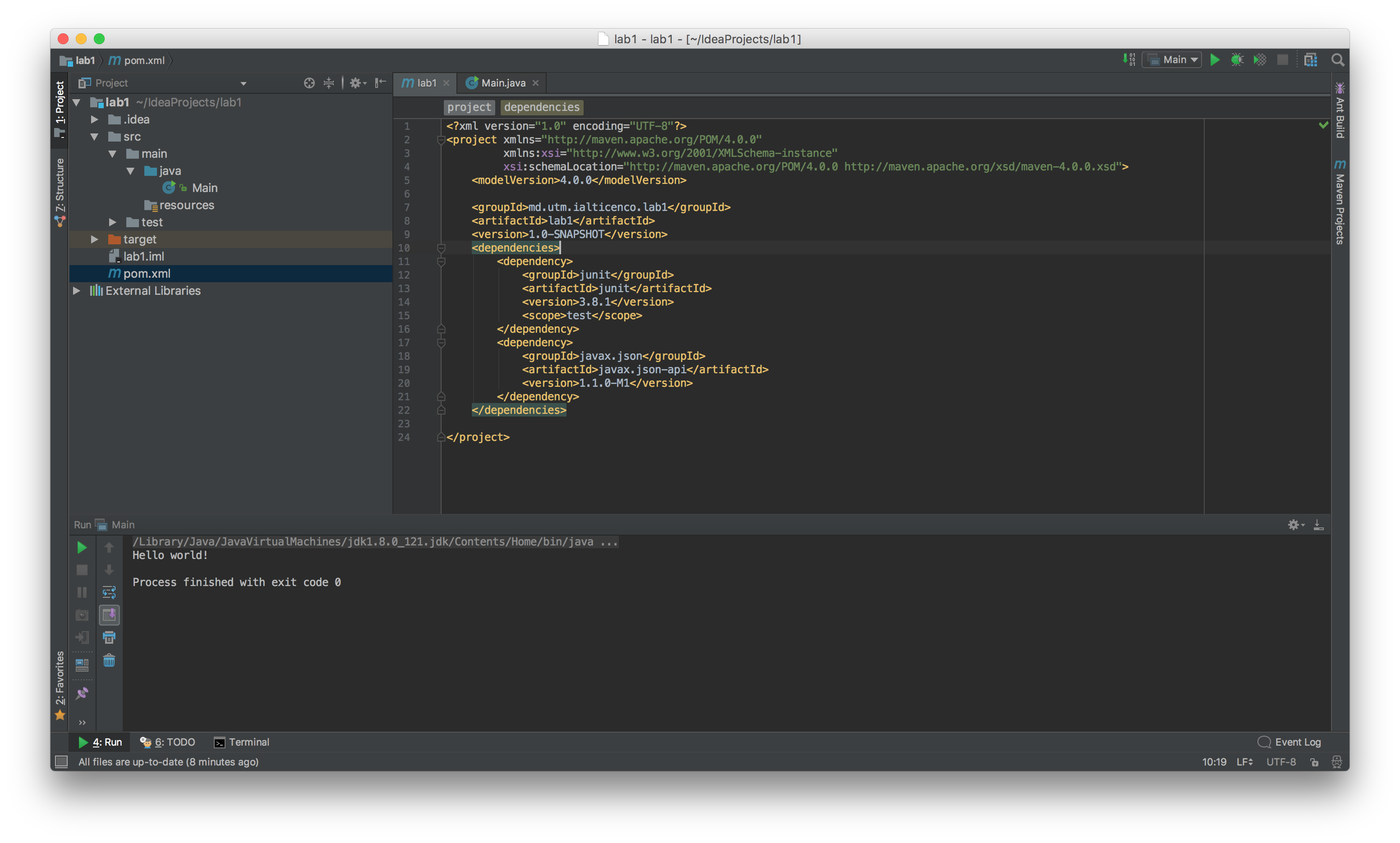


Рис. 8 – Измененный файл pom.xml

**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:

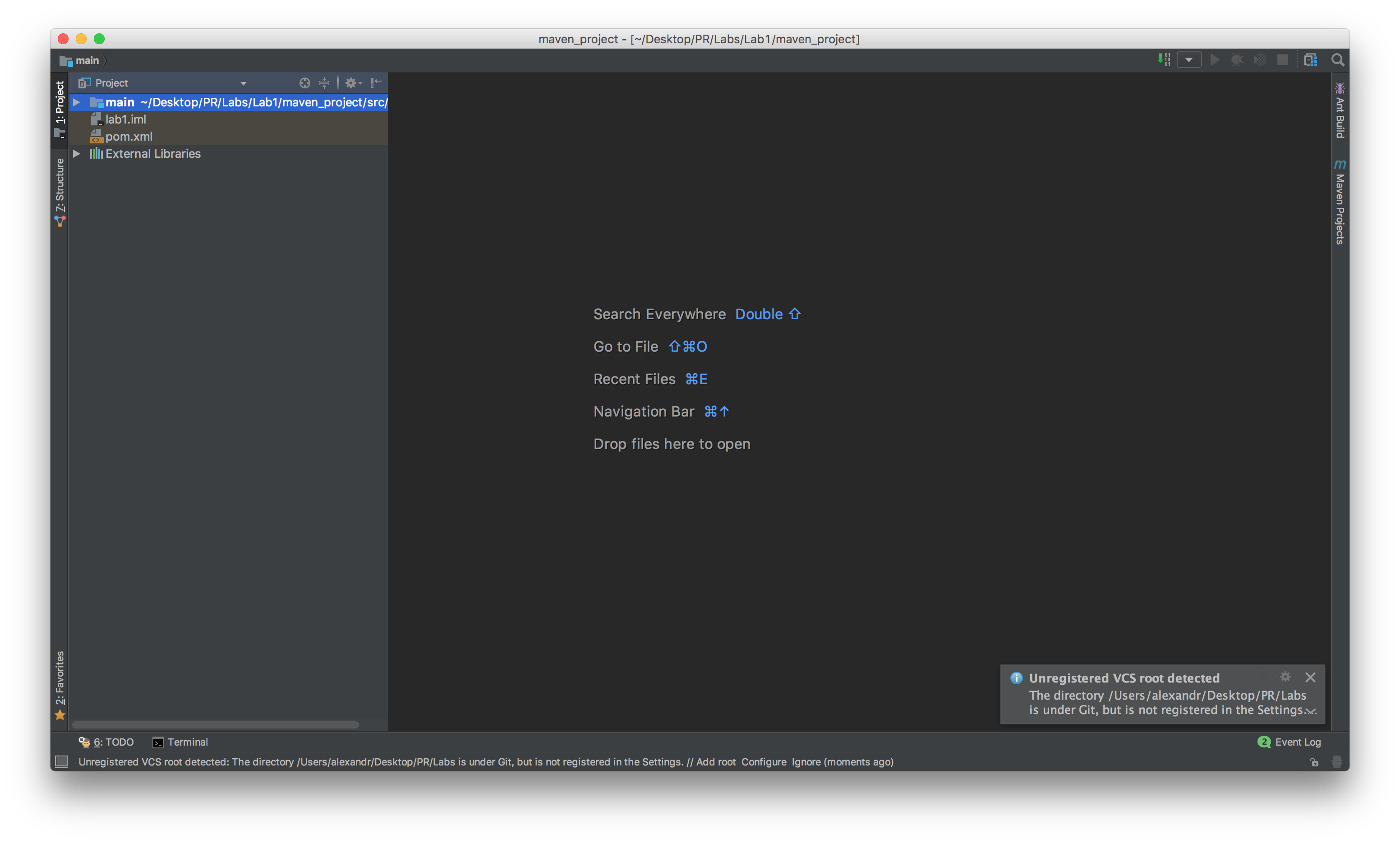


Рис. 9 – Intellij IDE обнаружение незарегистрированного VCS

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

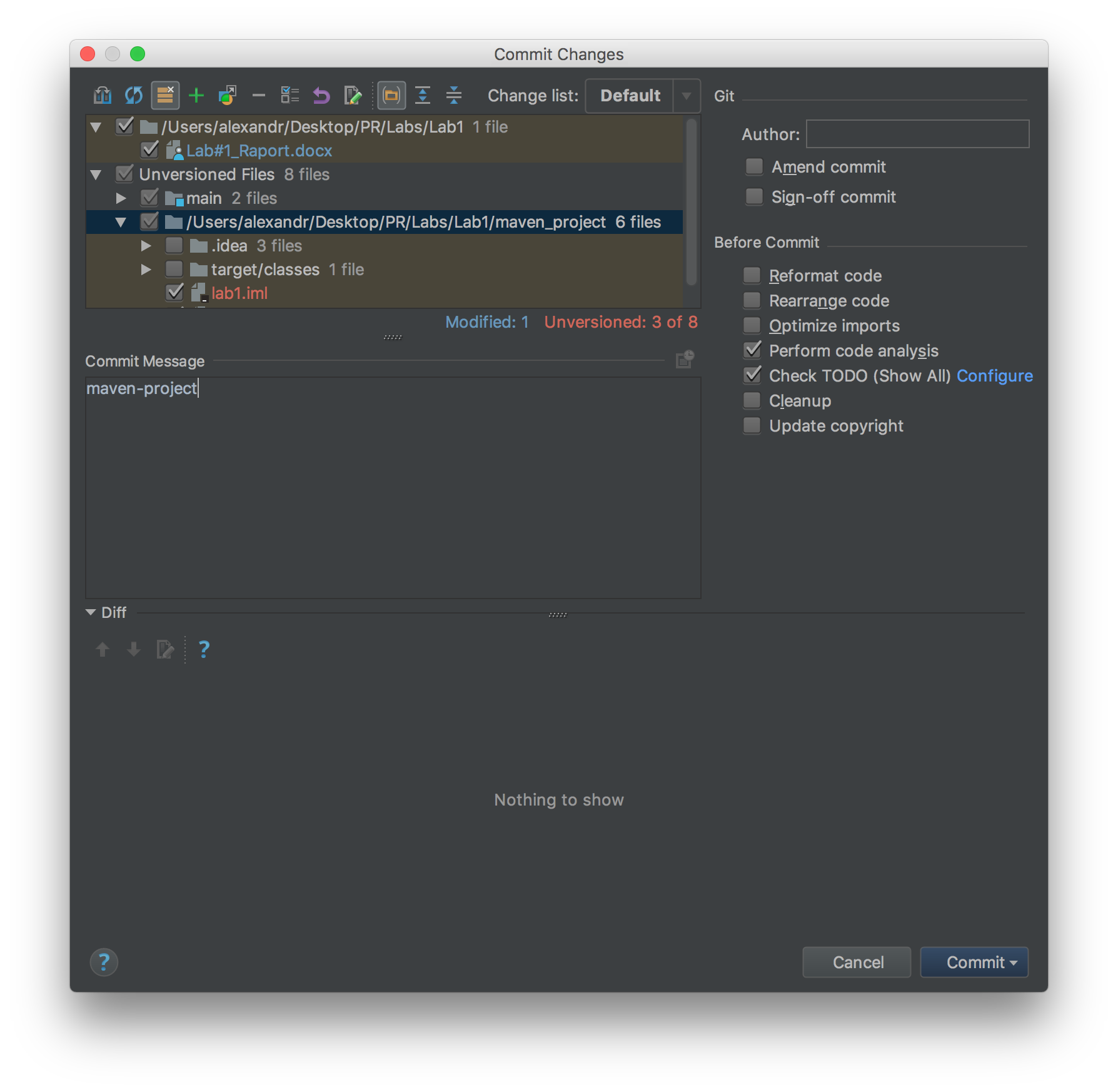


Рис. 10 – Осуществление GIT commit средствами Intellij IDE

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

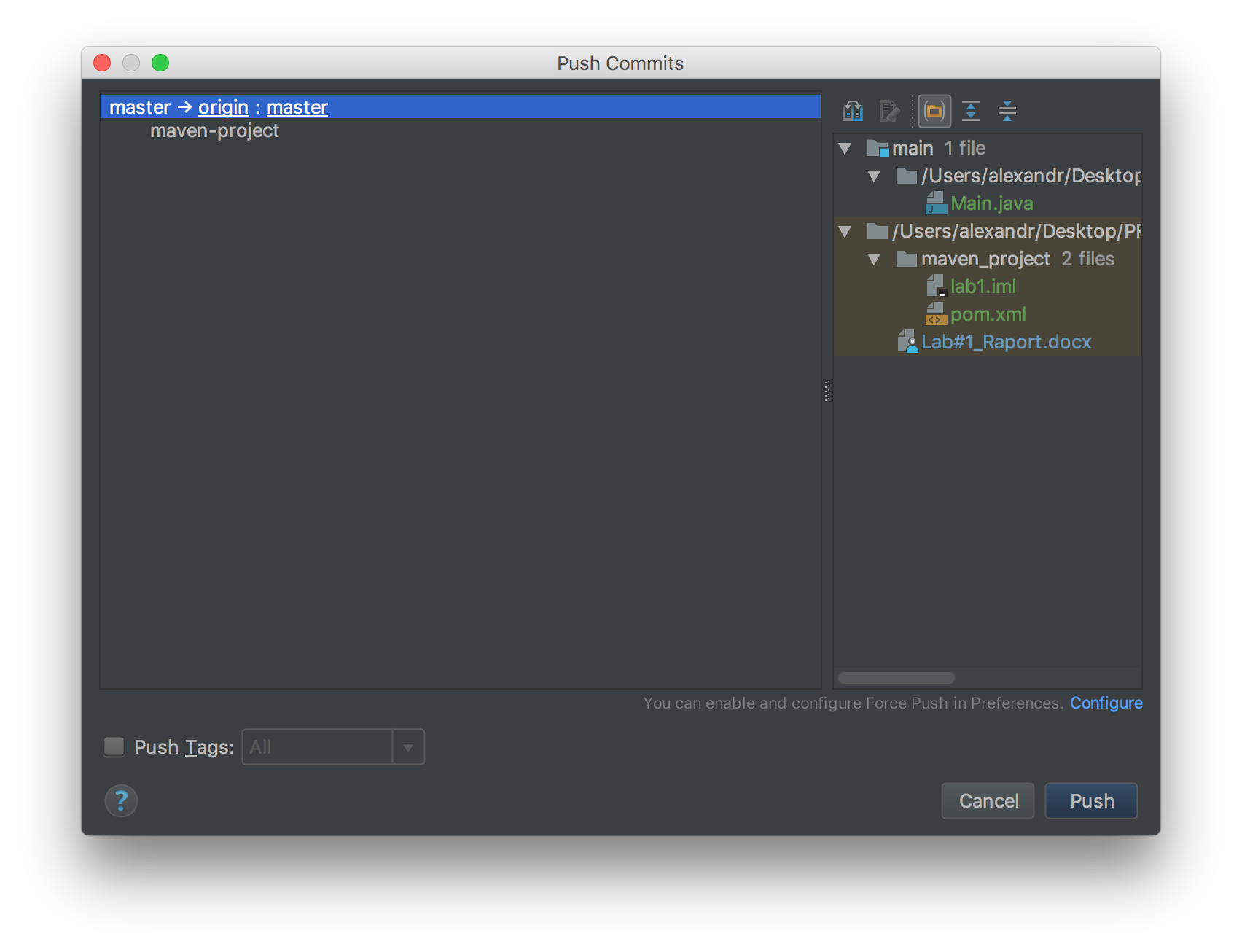


Рис. 11 – Отправка изменений на удаленный сервер через диалог Intellij IDEA

**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.