Git Repository

Github

Github - https://github.com - stellt freie, öffentliche Git Repositories mit 300 Mb Speicherplatz zu Verfügung. Unser Repository kann man über https://github.com/christophpurrer/mcm440-phone-app erreichen.

Die URL zu unserem Git Repository lautet:

- Für SSH >> git@github.com:christophpurrer/mcm440-phone-app.git
- Für HTTPS >> https://christophpurrer@github.com/christophpurrer/mcm440-phone-app.git
- Read-Only >> git://github.com/christophpurrer/mcm440-phone-app.git

Installation

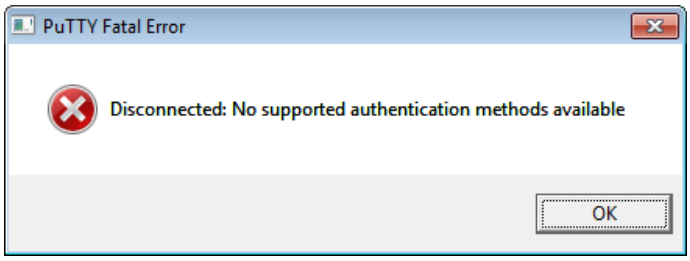
Git

Falls Git am System noch nicht vorhanden ist, kann man sich eine aktuelle Version von http://git-scm.com runterladen.

TortoiseGit

Nur für Windows gibt es zusätzlich noch das komfortable Tool ‚TortoiseGit' - http://code.google.com/p/tortoisegit, welches ein Ein- und Auschecken bequem vom Explorer aus ermöglicht.

TortoiseGit arbeitet intern mit PuTTY zusammen, um eine Authentifizierung vorzunehmen. Falls *‚Pageant'* nicht läuft bzw. noch kein gültiges PPK File zugewiesen wurde, dann erscheint folgende Fehlermeldung:



Die Pageant wird weiter unter erklärt.

SSH Public Key

Nach der Registrierung bei Github muss dem Repository mindestens 1 *‚SSH Public Key'* hinzugefügt werden. Hierzu startet man *die ‚Git Bash'*, welche im zuvor installierten Git Packet enthalten ist und folgt den Anweisungen von http://help.github.com/msysgit-key-setup.

Ich habe bereits für folgende Benutzer SSH Public Keys angelegt

- 1010455012@students.fh-hagenberg.at
- S0910629019@students.fh-hagenberg.at
- S1010455011@students.fh-hagenberg.at

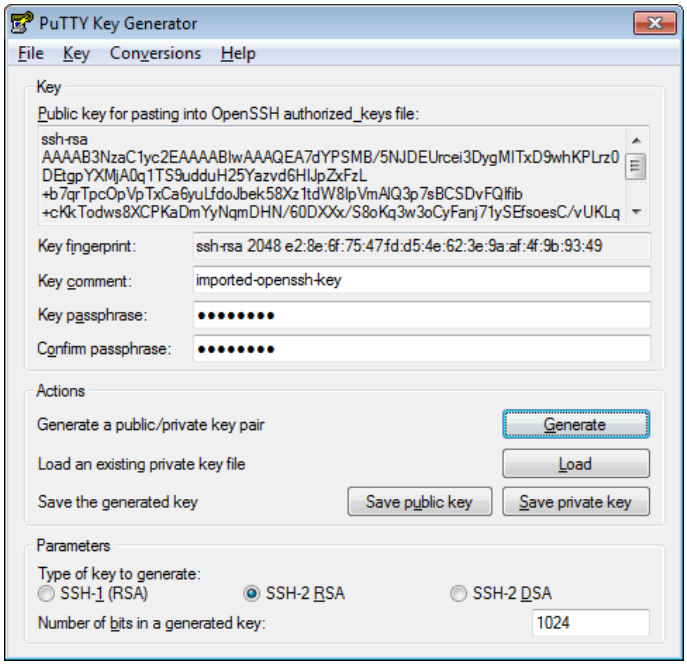
und bei den Account Einstellungen unter github.com hinzugefügt. Es daher nicht mehr nötig mit PuTTYGen *‚PuTTY Private Key Files'* zu erstellen.

Die SSH public / private Keys befinden sich anschließend im *USER_VERZEICHNIS/.ssh* am Betriebssystem.

PuttyGen

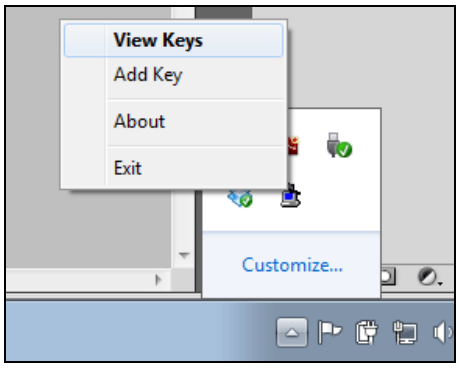
Um Git unter Windows komfortabel nutzen zu können braucht man noch ein *‚PuTTY Private Key File'* (ppk), dass man mit PuTTY Key Generator erstellen kann - http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html

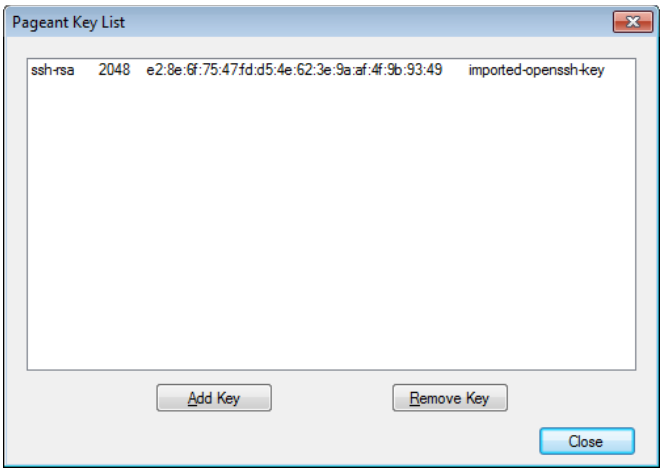
Einfach den zuvor generierten private Key laden, die zuvor generierte Passphrase eingeben und einmal als öffentlichen und privaten PuTTY Private Key exportieren.



Pageant

Um nicht für jede Git Transaction die Passphrase eingeben zu müssen verwendet man am besten Pageant, das im Hintergrund läuft und den / die Key(s) verwaltet - http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html.
Key beim Systemstart hinzufügen und Passphrase eingeben.





Git Tutorial

Nach erfolgreicher Installation kann man das Repository mit TortoiseGit oder per Command Line / Git Bash auschecken.

Command line mittels Git Bash

Repository auschecken

```
git clone git@github.com:christophpurrer/mcm440-phone-app.git
```

Datei / Verzeichnis hinzufügen und committen:

```
cd mcm440-phone-app/
touch testfile.txt
git add testfile.txt
git commit –m ‘my personal commit message’
git remote add origin git@github.com:christophpurrer/mcm440-phone-app.git
git push origin master
```

Datei / Verzeichnis löschen:

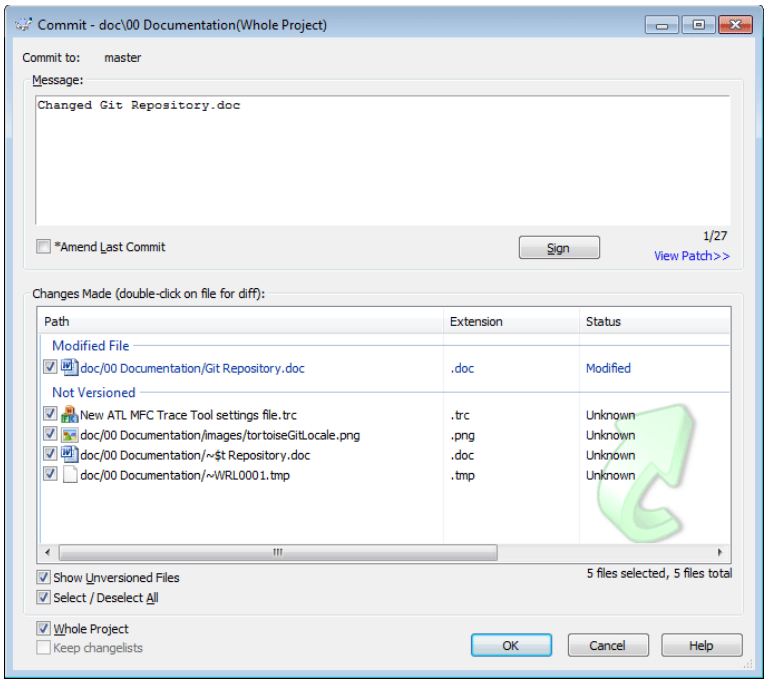
```
git rm testfile.txt
git commit –amend
```

Updates einchecken:

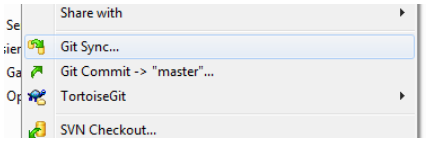
```
git pull
git push git@github.com:christophpurrer/mcm440-phone-app.git master:master
```

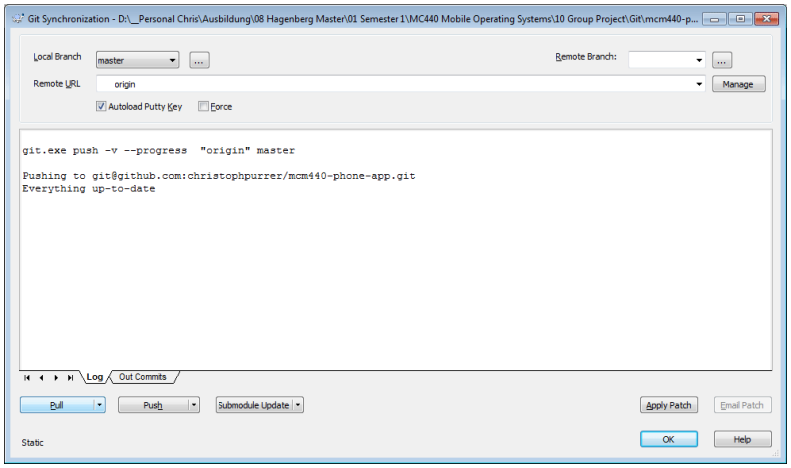
Git mit TortoiseGit unter Windows

Zuerst die gewünschten neuen oder modifizierten Daten zum lokalen Repository hinzufügen.



Anschließend ‚Git Sync …' wählen zuerst neue Daten vom Server pullen und dann die eigenen Daten zum Server pushen.





Linus Torvald über Git

http://www.youtube.com/watch?v=4XpnKHJAok8

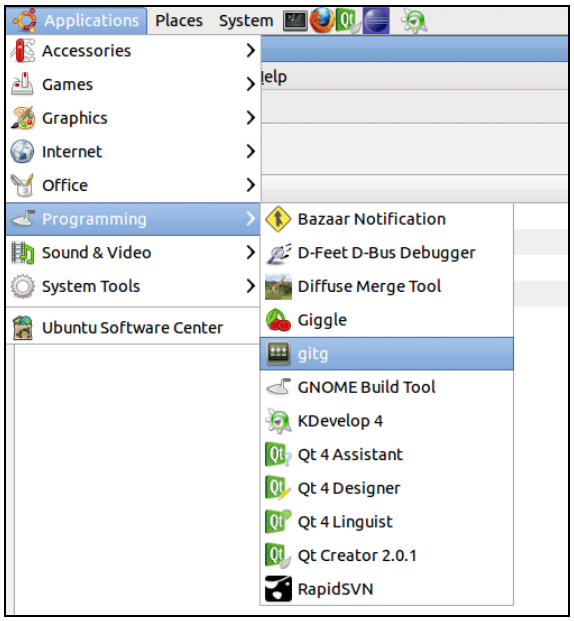
Gitg unter Linux

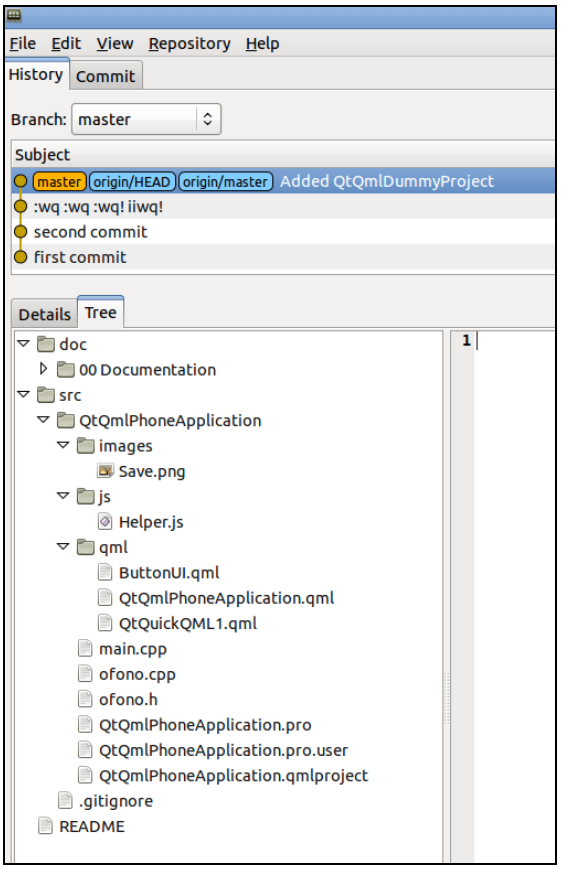
Git unter Linux installieren unter das *.ssh* Verzeichnis in das persönliche User Verzeichnis kopieren.

Project auschecken im User Verzeichnis auschecken:

```
git clone git@github.com:christophpurrer/mcm440-phone-app.git
```

Mit Gitg das Verzeichnis mcm440-phone-app öffnen …



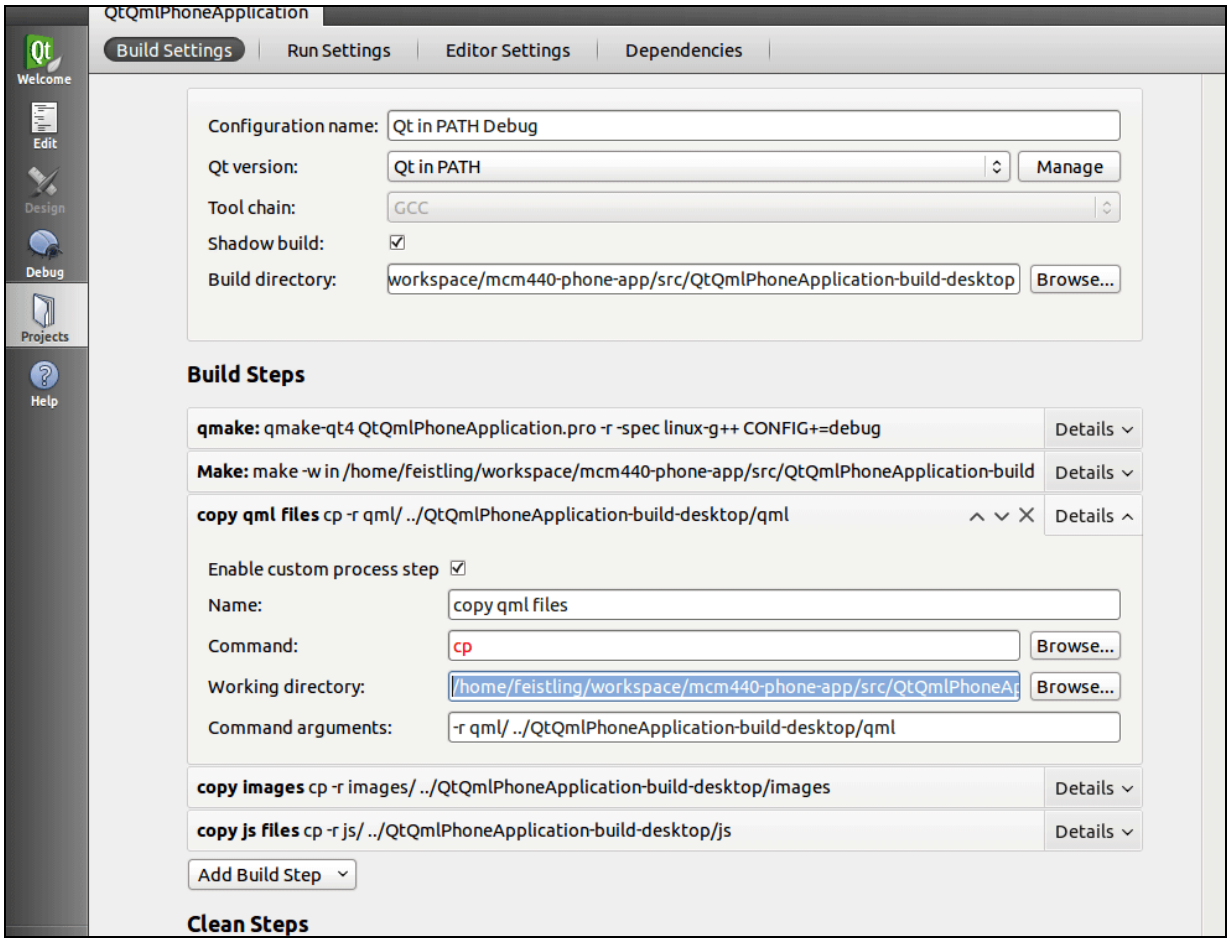


… und sich die Verzeichnisstruktur mal angucken.

Mittels QtCreator 2.x kann man das Programm als *‚Qt C++‘* Project – QtQmlPhoneApplication.pro – öffnen oder als *‚Qt Quick‘* Project – QtQmlPhoneApplication.qmlproject.

Der build Ordner QtQmlApplication-build-desktop wurde als *.gitignore* markiert.

3 Build Options im Qt C++ Project sorgen zur Zeit dafür das die 3 Ordner images, qml und js auch in den build Ordner kopiert werden. Leider ist hier das working directory noch absolut angegeben …


QtQmlPhoneApplication
Welcome
Edit
Design
Debug
Projects
Help
Build Settings
Run Settings
Editor Settings
Dependencies
Configuration name: Qt in PATH Debug
Qt version: Qt in PATH
Manage
Tool chain: GCC
Shadow build:
Build directory: workspace/mcm440-phone-app/src/QtQmlPhoneApplication-build-desktop
Browse...
Build Steps
qmake: qmake-qt4 QtQmlPhoneApplication.pro -r -spec linux-g++ CONFIG+=debug
Details
Make: make -w in /home/feistling/workspace/mcm440-phone-app/src/QtQmlPhoneApplication-build
Details
copy qml files cp -r qml/ ../QtQmlPhoneApplication-build-desktop/qml
Details
Enable custom process step
Name: copy qml files
Command: cp
Browse...
Working directory: /home/feistling/workspace/mcm440-phone-app/src/QtQmlPhoneAp
Browse...
Command arguments: -r qml/ ../QtQmlPhoneApplication-build-desktop/qml
copy images cp -r images/ ../QtQmlPhoneApplication-build-desktop/images
Details
copy js files cp -r js/ ../QtQmlPhoneApplication-build-desktop/js
Details
Add Build Step
Clean Steps