

Kurzreferenz Modul 183

Updates von Git einspielen

Nur nötig, wenn die Lehrperson Updates auf das Repository macht, welche Sie gerne übernehmen möchten.

<https://gitlab.com/borisdäppen/School-Security-Application>

cd School-Security-Application	# zu Code Repository wechseln
git config --global user.name 'Mein Name'	# nur einmalig:
git config --global user.email name@host.ch	# User-Infos angeben
git branch	# prüfen ob auf Branch «student»
git add -u	# alle Ihre Änderungen markieren
git commit -m 'Abschluss Arbeitsblatt X'	# Ihre Änderungen «speichern»
git checkout master	# zu Haupt-Branch wechseln
git pull origin master	# Updates herunterladen
git checkout student	# zurück zu Ihrem Branch
git merge master	# Updates in Ihren Branch mergen

Eigene Änderungen rückgängig machen / Datei auf «letzten Zustand» zurück setzen:

git checkout code/client/ziu

Server bedienen

vmLP1 – Änderungen im Server-Code auf den Server verteilen und Server direkt starten:

```
cd code/server
bash deploy.sh
```

vmLP1 – Server auf Entwickler-PC direkt ab Source starten

```
cd code/server
perl -Ilib bin/fulla
```

vmLS3 – Server manuell bedienen

fulla	# Server manuell im Vordergrund starten (Logmeldungen sichtbar)
fulla bg	# Server im Hintergrund als Daemon starten (nichts mehr sichtbar)
pgrep -fa fulla	# Prozess suchen
pkill -F pid.fulla	# Prozess beenden
tail -f fulla.*	# Logfiles live verfolgen

Client bedienen

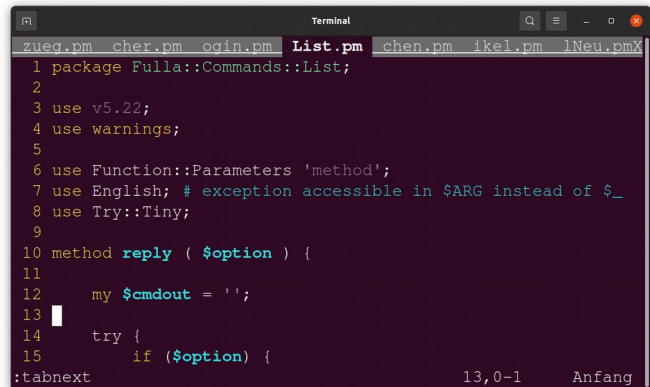
ziu --help	# Anleitung anzeigen
ziu login admin anna	# Auf Instanz von Entwickler-PC verbinden (localhost)
FULLAIP=192.168.220.13 ziu list	# Nachricht an andere IP senden
export FULLAIP=192.168.220.13	# IP-Einstellung dauerhaft festlegen
cd /code/client; bash deploy.sh	# Client-Code in System installieren
cd /code/client; perl ziu list	# Client ab Source starten

Editoren fortgeschritten einrichten

Für den **VIM** existiert eine Projekt-konfiguration auf dem Repository.

Folgende Features werden damit eingerichtet:

- Tabulatoren für mehrere Dateien:
Space: Nach rechts blättern
CTRL+Space: Nach links blättern
- Highlighting für Projekt-Syntax:
method add (\$banana)
- Einfaches Seiten-Scrollen:
Backspace: Nach oben blättern
Enter: Nach unten blättern



```
1 package Fulla::Commands::List;
2
3 use v5.22;
4 use warnings;
5
6 use Function::Parameters 'method';
7 use English; # exception accessible in $ARG instead of $_
8 use Try::Tiny;
9
10 method reply ( $option ) {
11
12     my $cmdout = '';
13
14     try {
15         if ($option) {
```

Zur Aktivierung müssen Sie lediglich einige Datei-Links auf Ihrem System setzen:

```
mkdir -p .vim/after/syntax
```

```
ln -s ~/School-Security-Application/dev/vim/after/syntax/perl.vim \
    .vim/after/syntax/perl.vim
```

```
ln -s ~/School-Security-Application/dev/vim/vimrc .vimrc
```

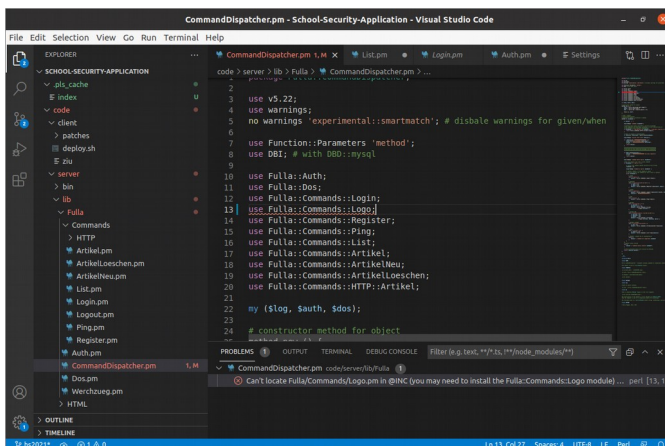
Danach können Sie mehrere Dateien gleichzeitig in Tabulatoren öffnen:

```
cd School-Security-Application/code/server
```

```
vim -p bin/fulla lib/Fulla/*
```

```
vim -p lib/Fulla/Commands/*
```

```
vim -p $(find lib -type f)
```



Für **Visual Studio Code** gibt es einen Perl Language Server.

Unter <https://marketplace.visualstudio.com/items?itemName=FractalBoy.pls> ist eine Anleitung.

Die groben Schritte:

1. Das Perl-Paket PLS installieren:

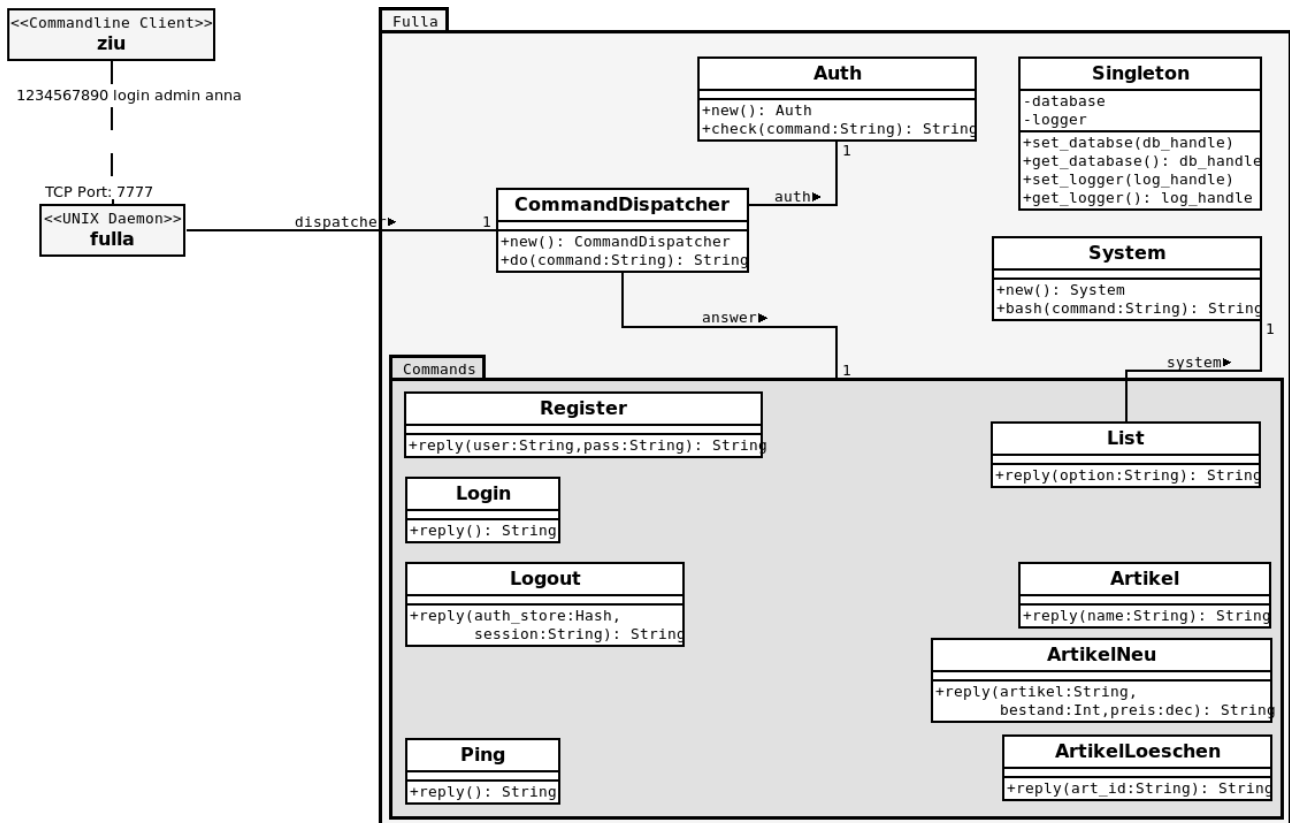
```
sudo apt install cpanminus
```

```
sudo cpanm PLS
```
2. Die in VSCode die Erweiterung «FractalBoy» installieren.

Alle anderen Editoren wie sublime, atom, nano, emacs, ... sind für das Modul auch gut genug!

Code-Übersicht

Im Repository hat es unter `/dev` eine Übersicht über die Pakete bzw. Klassen:



Das Diagramm entspricht weder einem strengen Standard, noch ist es komplett. Es zeigt aber die wichtigsten Zusammenhänge auf einen Blick.