Installation des M1 Macbooks

1. Inhaltsverzeichnis

1. EINLEITUNG

In diesem Skript wird erklärt, wie das MacBook für einen neuen Benutzer so eingerichtet werden kann, dass es für die grundsätzlichen Aufgaben im ersten Jahr der Fakultät 73 reibungslos funktioniert.

Das Skript ist für ein M1 MacBook mit MacOS-Monterey geschrieben und KEINE offizielle Arbeit von Volkswagen oder der Fakultät73. Daher gibt es keine Garantie, dass alles 100% funktioniert. Am Ende dieses Skripts ist ein Quellenverzeichnis, in welchem bei Bedarf nach Lösungen für Probleme nachgeschaut werden kann.

Zscaler App

Achte bitte darauf, dass dein MacBook als Proxy Software den Zscaler benutzt. Ist dies nicht der Fall, wird das meiste in diesem Skript nicht funktionieren.

2. EINRICHTUNGSOPTIONEN

Es gibt zwei Möglichkeiten, das MacBook einzurichten:

1. Semi-Automatisch
   Es wird ein Skript ausgeführt, welches Java, Maven, Homebrew installiert und allgemeine Proxy Einstellungen ausführt.
2. Manuell
   Alle Einstellungen und Installationen müssen selbst durchgeführt werden.

2. SEMI-AUTOMATISCHE INSTALLATION

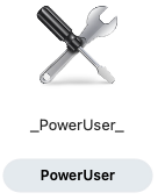
In dem Ordner, in dem du dieses Hilfe-Skript gefunden hast, befindet sich auch eine Datei namens: *install.sh*
Dies ist ein Shell-Skript, welches alle Anweisungen für einen Teil der Installation enthält.
Du verwendest es wie folgt:

3. Aktiviere den PowerUser-Modus:

    a. Öffne die SelfService-App auf deinem MacBook.

    b. Suche und aktiviere den _PowerUser_

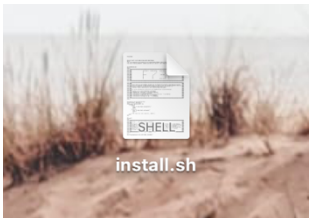
    

    c. Warte, bis dieser aktiv ist und in der Statusleiste erscheint.

4. Kopiere das Skript *install.sh* an einen beliebigen Ort auf deinem System.

    a. In unserem Beispiel auf den Desktop:

5. Öffne ein Terminal und navigiere zu deinem gewählten Pfad.

```
cd /Users/DEINE_USER_ID/Desktop
```

WICHTIG:
Das Terminal muss vor dem Öffnen komplett geschlossen sein. Wenn es noch einen Punkt unter seinem Symbol hat, öffne es und drücke CMD + Q zum vollständigen Beenden. Sonst ist der PowerUser nicht für dieses Programm aktiv.

Richtig:  Falsch: 

6. Mache das Skript ausführbar und führe es aus:

```
sudo chmod +x install.sh
sh install.sh
```

7. Folge den Anweisungen im Terminal. Halte dazu dein Passwort bereit, welches du zum Anmelden an das MacBook benutzt. Das Skript wird einige Dinge abfragen. Wenn du nicht sicher bist, was du tun sollst, schau in Kapitel 4. Dort sind alle Operationen des Skripts manuell beschrieben.

4. MANUELLE INSTALLATION

Sollte das Skript nicht funktionieren oder solltest du es nicht benutzen wollen, kannst du auch alles selbst einstellen und installieren. Beachte bitte, dass aufgrund des Umfangs hier nicht alles im Detail erklärt werden kann.

Das Terminal (Bash) für den Proxy vorbereiten

Dein MacBook (MacOS Monterey) nutzt eine sogenannte *.zshrc*-Datei, in der Links und Anweisungen für die Bash abgelegt werden können. Damit die Bash durch den VW-Proxy (also mit eingeschaltetem Zscaler) auch das Internet erreichen kann, muss hier auf den Proxy verwiesen werden.

Öffne ein Terminal und gebe ein:

```
nano ~/.zshrc
```

Die Datei wird in „nano" (einem Terminal Texteditor) geöffnet. Füge jetzt folgende zwei Zeilen hinzu:

```
export http_proxy=http://127.0.0.1:9000
export https_proxy=http://127.0.0.1:9000
```

Diese Zeilen bewirken, dass alle Anfragen ins Internet zuerst an diese *http*-Adresse umgeleitet werden. (127.0.0.1 ist dein „localhost") Dort liegt eine *pac*-Datei, die automatisch deine Anfrage an die richtige Adresse leitet.

HINWEIS:
Wenn du es so wie auf folgendem Screenshot machst, wird das Terminal bei jedem Start prüfen, ob die Zscaler App läuft und den Proxy automatisch setzen.

```
if pgrep -x "Zscaler" > /dev/null; then
    echo "Zscaler running - SET PROXY."
    export http_proxy=http://127.0.0.1:9000
    export https_proxy=http://127.0.0.1:9000
else
    echo "Zscaler not running - NO PROXY SET."
fi
```

Speichere deine Änderung mit folgenden Tasten:
Control + X → Y → Enter

WICHTIG:
Änderungen an der *.zshrc*-Datei werden IMMER erst aktiv, wenn du einen der beiden folgenden Befehle ausführst:

```
source ~/.zshrc
. ~/.zshrc
```

Alternativ zu den Befehlen kannst du das Terminal auch neu starten.
(*CMD + Q*, dann neu öffnen)

Homebrew installieren

Homebrew ist ein Paket-Manager für MacOS. Falls du bereits Linux kennst – es ist vergleichbar mit apt, dnf oder yum. Mit Homebrew lassen sich leicht Anwendungen installieren oder deinstallieren. Weiteres dazu findest du hier.

Öffne wieder ein Terminal als PowerUser und gib folgenden Befehl ein (achte darauf, dass er in EINER Zeile, ohne Umbruch steht):

```
/bin/bash -c "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

Homebrew wird nun installiert, das kann eine Weile dauern. Danach bearbeite wieder die *.zshrc*-Datei und füge folgende Zeile hinzu:

```
eval \$(/opt/homebrew/bin/brew shellenv)
```

Speichere deine Änderung und aktualisiere dein Terminal, wie in der roten Box oben beschrieben.
Ob alles funktioniert hat, testen wir direkt. Führe folgenden Befehl aus, um Homebrew & die Repositorys upzudaten:

```
brew update
```

Hinweis:
Programme, die mit Homebrew installiert werden, landen im Normalfall hier:
/opt/homebrew/Cellar/
Es gibt aber auch Programme, die an anderen Orten abgelegt werden. Java ist solch ein Fall.

Java Development Kit (JDK) installieren

Um Software in Java entwickeln zu können, brauchst du eine JDK – das sogenannte Java Development Kit. Es gibt verschiedene Versionen und Forks der JDK. Die Fakultät73 setzt sehr viel auf Java 8 (1.8) und zudem auf ein zusätzliches Package, welches JavaFX heißt. JavaFX ist nicht mehr Teil der Standard JDK, einige Java Versionen beinhalten es aber trotzdem noch. Aus diesem Grund installieren wir ein Java 8 aus diesem Repository: bell-sw/homebrew-liberica (github.com)

Dazu öffne wieder ein Terminal und gebe folgende Befehle ein:

```
brew tap homebrew/cask-versions
brew install --cask liberica-jdk8-full
```

Wenn du mehr als nur Java 8 installieren möchtest, kannst du auf der Website nach weiteren Versionen suchen und diese installieren. Beachte aber, es kann immer nur eine Java-Version für das MacOS als „Default" gesetzt werden. Für die Entwicklung innerhalb der IDE ist dies aber irrelevant.

Aktualisiere nach der Installation dein Terminal und prüfe ob Java korrekt gesetzt ist:

```
java -version
```

Hinweis:
Solltest du mehr als eine Java Version installieren wollen, hier ein netter Workaround, wie man es lösen kann:
Nehmen wir an, es sind Java 8 und Java 18 von liberica-Repo installiert.
Öffne ein Terminal und öffne dort wieder die *.zshrc*-Datei. Füge dann folgende Zeile(n) an (Achte darauf, die Zeilenumbrüche zu entfernen!):

```
alias j8="export JAVA_HOME=/Library/Java/JavaVirtualMachines/liberica-jdk-8-full.jdk/Contents/Home; java -version"

alias j18="export JAVA_HOME=/Library/Java/JavaVirtualMachines/liberica-jdk-18-full.jdk/Contents/Home; java -version"
```

Speichere die Änderung und aktualisiere anschließend das Terminal. Jetzt kannst du die „Default"-Version ganz einfach wechseln, indem du *j8* bzw. *j18* eintippst. Achte auf die korrekten Pfade! Du kannst auch die Aliase von *j8* / *j18* in einen belieben anderen Wert ändern oder weitere Versionen so hinzufügen.

Unsere .zshrc-Datei sieht in unserem Beispiel nun so aus:

```
if pgrep -x "Zscaler" > /dev/null; then
    echo "Zscaler running - SET PROXY."
    export http_proxy=http://127.0.0.1:9000
    export https_proxy=http://127.0.0.1:9000
else
    echo "Zscaler not running - NO PROXY SET."
fi
eval $(/opt/homebrew/bin/brew shellenv)
alias j8="export JAVA_HOME=/Library/Java/JavaVirtualMachines/liberica-jdk-8-full.jdk/Contents/Home; java -version"
alias j18="export JAVA_HOME=/Library/Java/JavaVirtualMachines/liberica-jdk-18-full.jdk/Contents/Home; java -version"
```

Maven & Eclipse installieren

Jetzt fehlt noch Maven. Dies ist ein Build- und Packagemanager für Java. Maven kann ganz einfach installiert werden mit:

```
brew install maven
```

Ob das funktioniert hat, verrät der Befehl:

```
mvn -version
```

Kommt hier kein Fehler, sondern eine Version, ist Maven nutzbar.
Maven wird immer deine „Default"-JDK benutzen.

Dazu brauchst du noch eine IDE, zum eigentlichen Code schreiben. Die Fakultät73 setzt hier auf Eclipse. Diese ist eine Open Source IDE, die ebenfalls leicht über Homebrew installierbar ist:

```
brew install --cask eclipse-java
```

Eclipse erscheint danach als Symbol in deinem Launchpad.

Maven für den Proxy anpassen

Damit Maven durch den VW-Proxy mit dem Internet kommunizieren kann, muss dieser konfiguriert werden. Dazu sind folgende Schritte notwendig:

Öffne die folgende Datei:

/Users/DEINE_USER_ID/.m2/settings.xml

Sollte der .m2-Ordner und/oder die Datei fehlen, erstelle beides einfach. Maven wird sie erkennen, sowie sie da sind.
Füge in die Datei den Text der folgenden Seite ein (beachte, dass du den Pfad zum *<localRepository>* unbedingt mit deiner User ID ergänzt!):

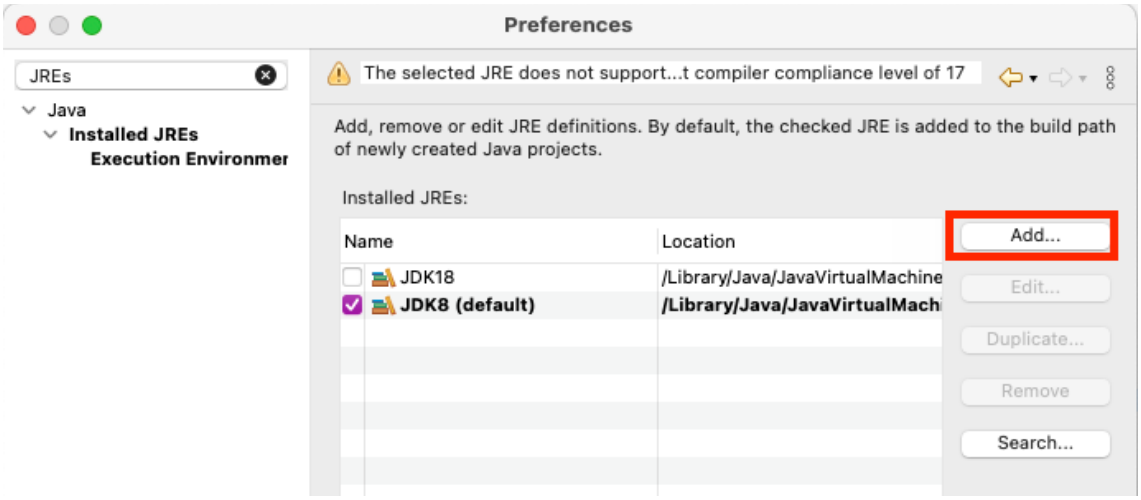
```
<settings>

<!-- localRepository
| The path to the local repository maven will use to store artifacts.
| Default: ~/.m2/repository
| <localRepository>/path/to/local/repo</localRepository> -->

<localRepository>DEINE_USER_ID/.m2</localRepository>

<!-- offline
| Determines whether maven should attempt to connect to the
| network when executing a build.
| This will have an effect on artifact downloads, artifact
| deployment, and others.
| Default: false -->

<offline>false</offline>

<proxies>
    <proxy>
        <id>optional</id>
        <active>true</active>
        <protocol>http</protocol>
        <host>127.0.0.1</host>
        <port>9000</port>
    </proxy>
</proxies>

</settings>
```

Hinweis:
Hier endet das automatisierte *install.sh* – Skript.

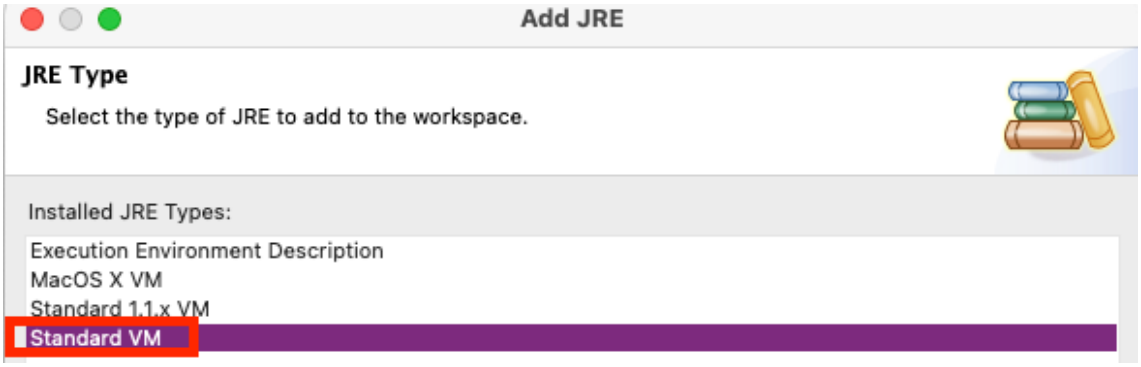
5. ECLISPE FERTIG EINRICHTEN

Um anschließend mit Eclipse problemlos arbeiten zu können, muss die IDE noch fertig eingerichtet werden. Nachdem du einen Workspace-Pfad gewählt hast, kann es passieren, dass Eclipse die zuvor installierten JDKs nicht findet. Diese können manuell eingestellt werden:

Klicke oben links auf *Eclipse → Preferences → Suche in diese Fenster nach „JREs“*
Sind hier deine JDKs nicht aufgelistet (in diesem Beispiel sind sie es), klicke auf *Add...*

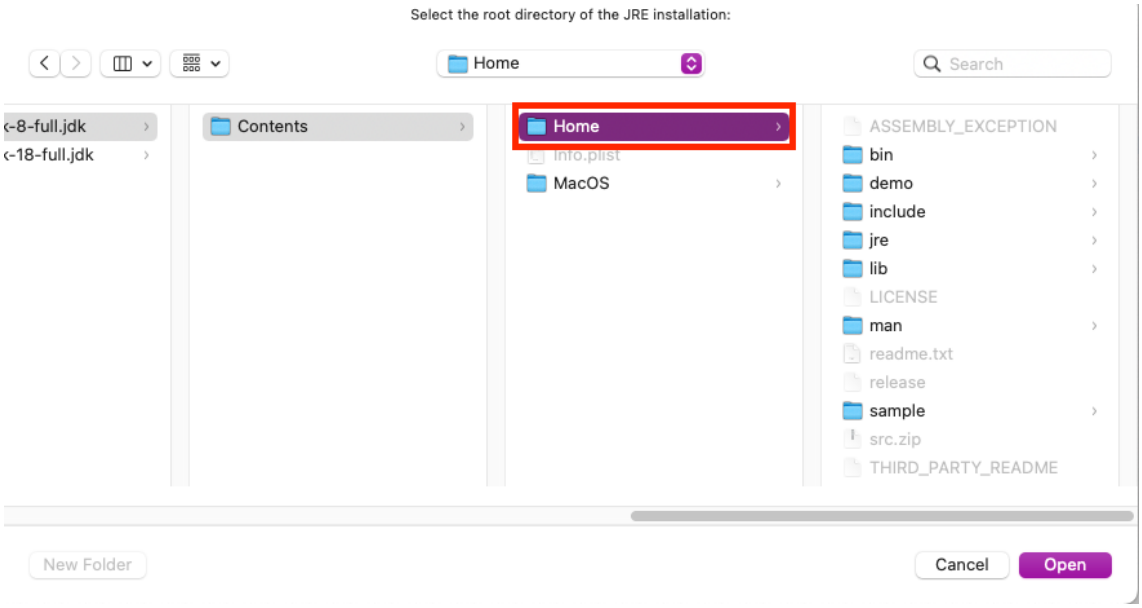


Wähle *Standard VM* und danach *Weiter / Next*

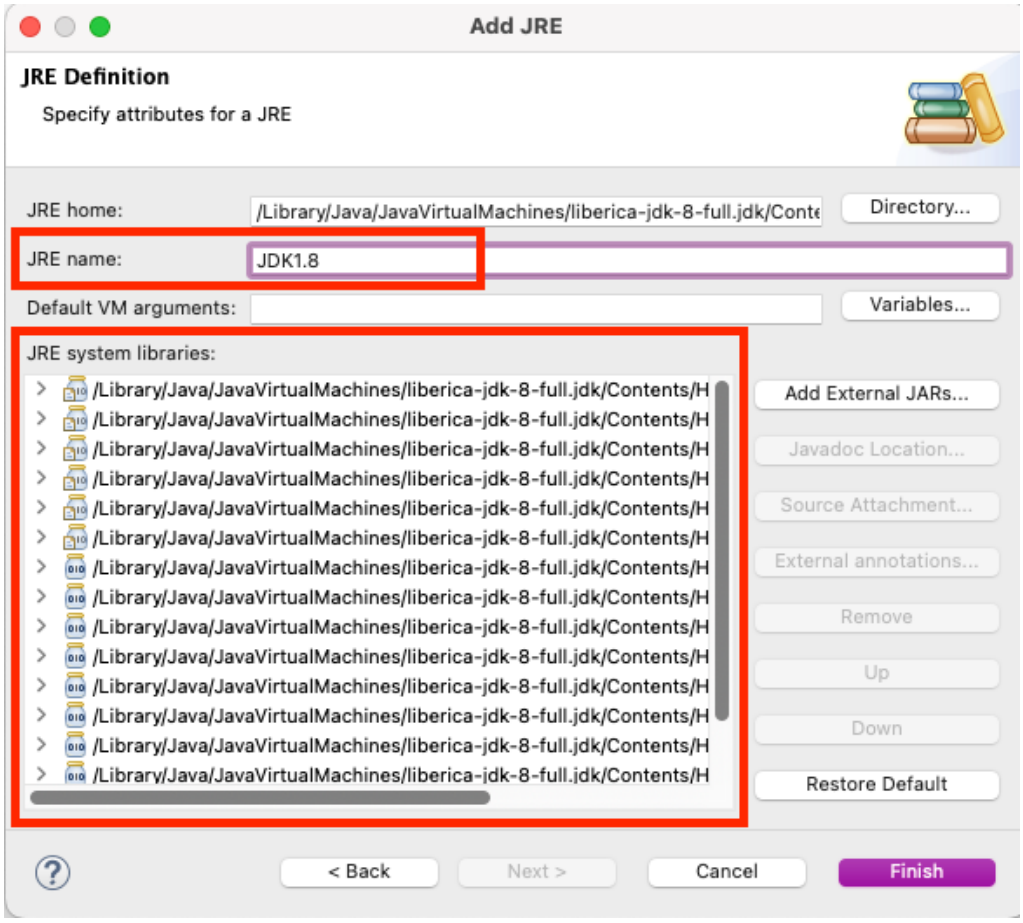


Klicke auf *Directory...*

Suche den Pfad zum *Home*-Verzeichnis der JDK. Im Normalfall findest du es unter: */Library/Java/JavaVirtualMachines/JDK-NAME/Contents/Home*

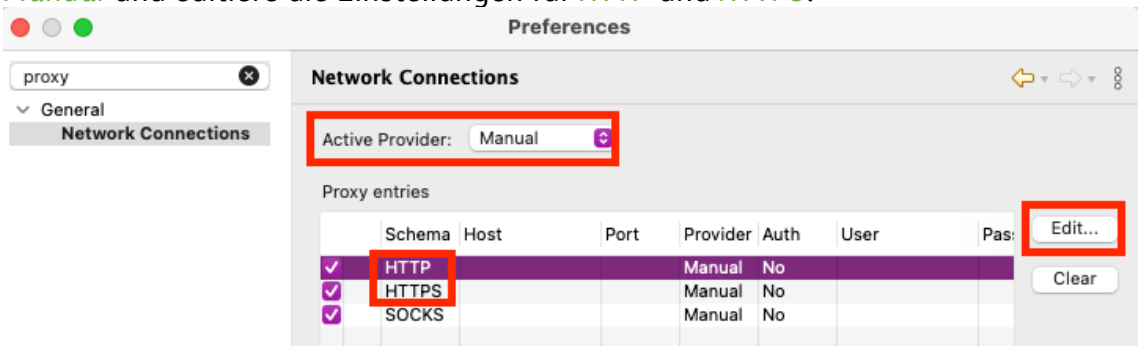


Hast du das korrekte Verzeichnis gewählt wird Eclipse die Packages der JDK erkennen und sie unten anzeigen. Vergib am besten noch einen Namen, damit du die JDK später wiedererkennst:

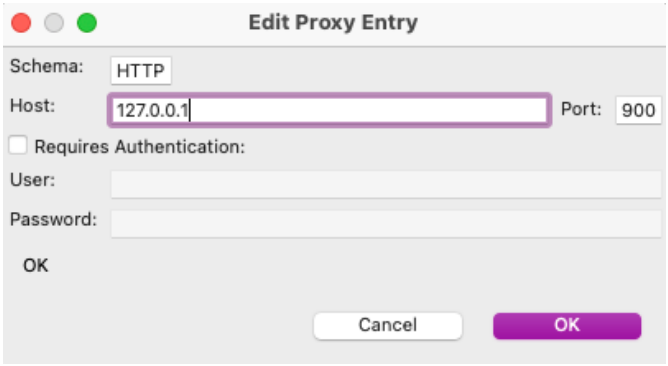


Wiederhole diesen Vorgang, bis du alle JDKs, die du in Eclipse benutzen möchtest, hinzugefügt hast.

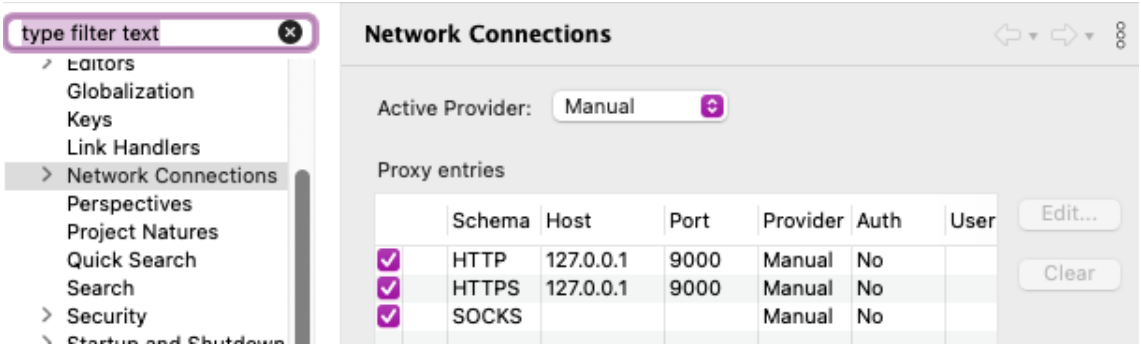
Jetzt musst du noch den Proxy einrichten. Dazu wieder in die *Preferences* gehen und in dem Suche Fenster nach *Network Connections* suchen. Wähle oben im Dropdown *Manual* und editiere die Einstellungen für *HTTP* und *HTTPS*:



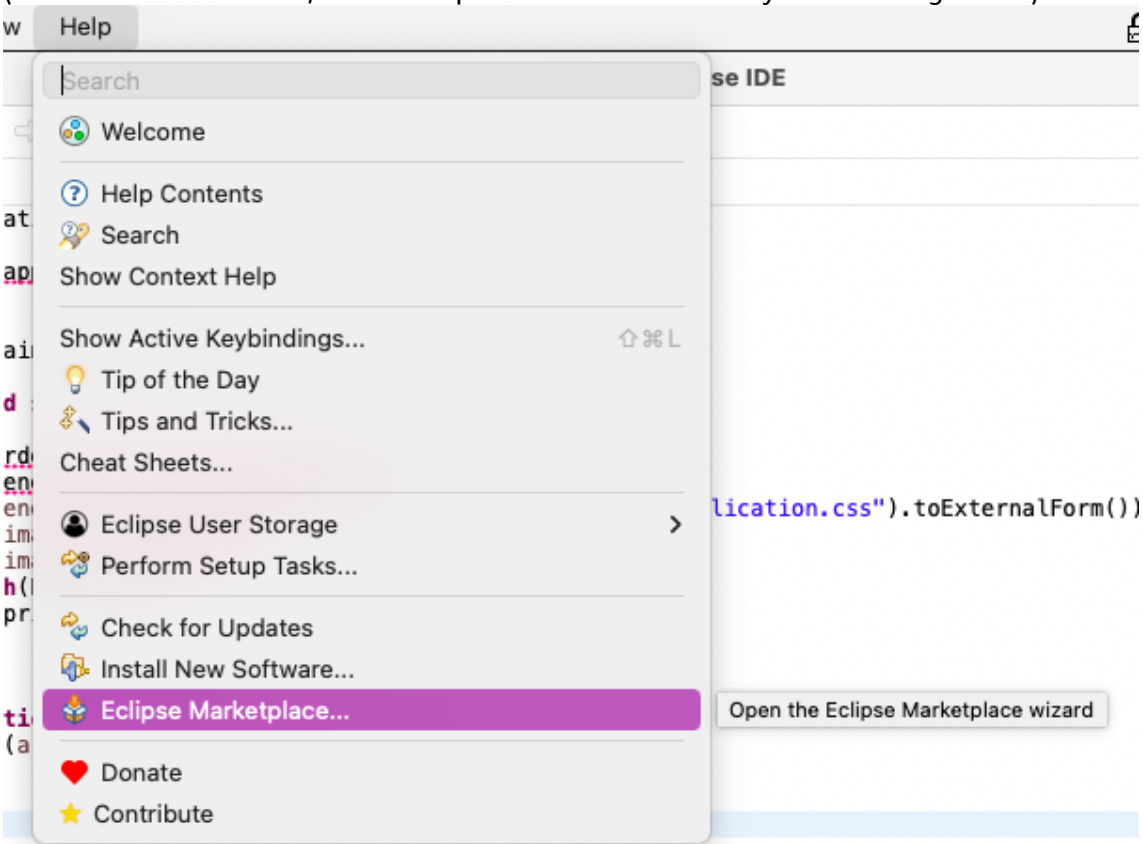
Beide Schemas bekommen folgende Einträge (Alle anderen Felder bleiben leer.):
Hots: 127.0.0.1
Port: 9000



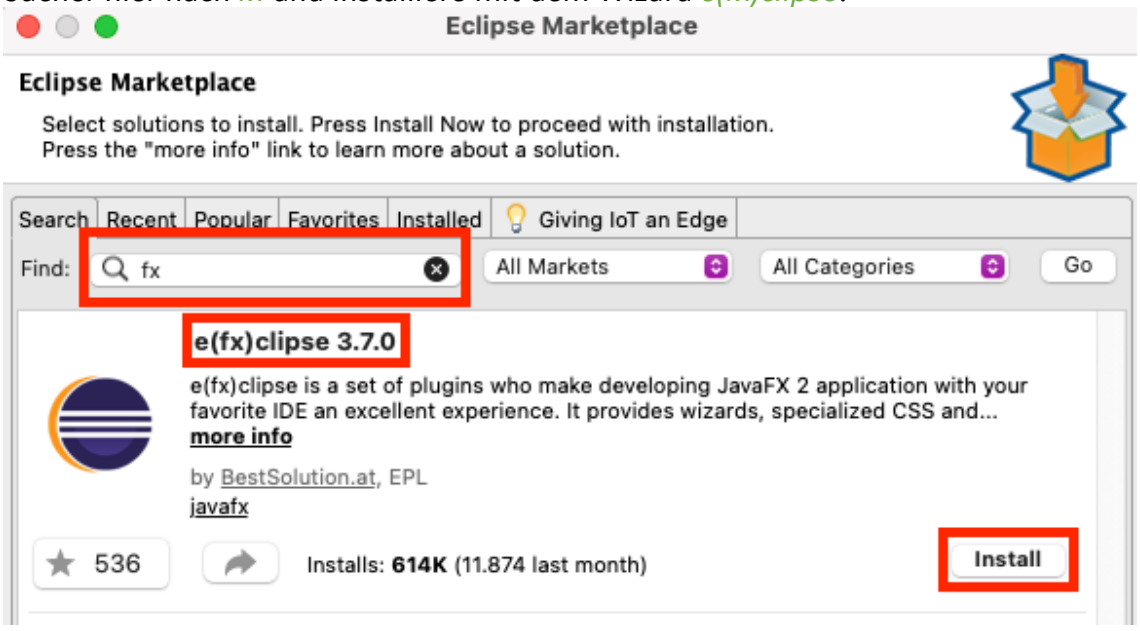
Am Ende sollte es so aussehen:



Ob das alles funktioniert hat, testen wir jetzt direkt. Du brauchst noch ein wichtiges Plugin zur Entwicklung von JavaFX. Um Plugins zu installieren, öffne den Marketplace (Dieser öffnet sich nur, wenn Eclipse die korrekten Proxy-Einstellungen hat):



Sucher hier nach *fx* und installiere mit dem Wizard *e(fx)clipse*:



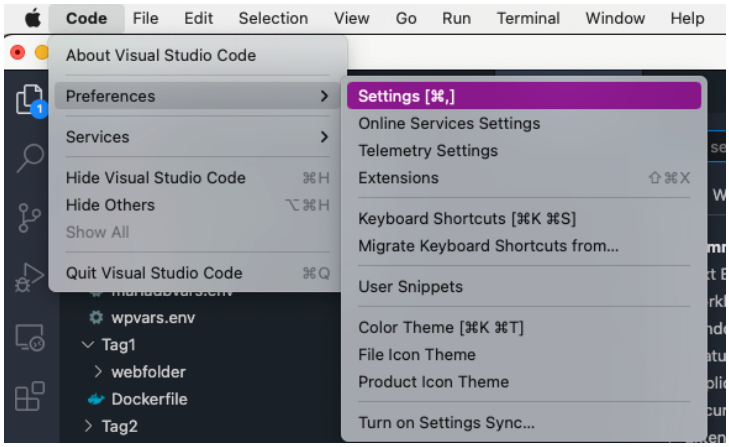
Damit sollte Eclipse fertig und einsatzbereit für dich sein.

6. WEITERE SOFTWARE

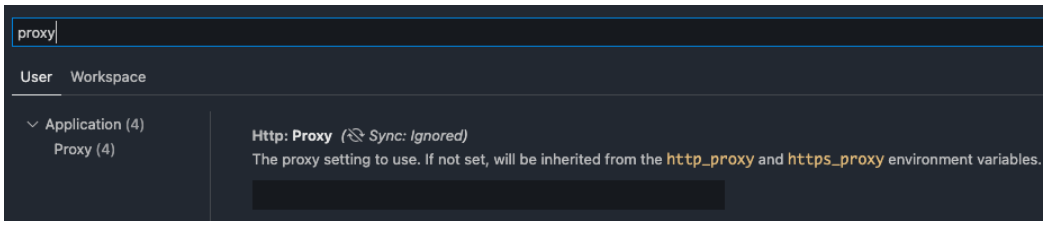
Du wirst natürlich nicht nur Eclipse in der Fakultät73 benutzen. Es gibt eine Vielzahl von Software, die du brauchst oder welche du selbst bevorzugst. Im Folgenden zeige ich noch ein paar Programme, die ich oft benutze und wie diese auf den Proxy konfiguriert werden.

VS Code:

Visual Studio Code kannst du aus dem SelfService auf deinem MacBook installieren. Konfiguriere es so, wie du es für richtig empfindest. Proxyeinstellungen sind hier nicht nötig. Achte nur darauf, dass nichts voreingestellt ist. VS Code wird sich die Einstellungen automatisch aus der *.zshrc*-Datei ziehen.

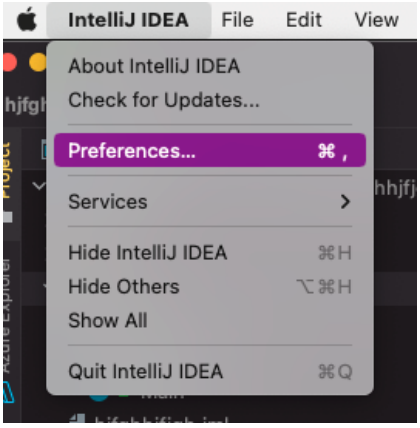


In den Einstellungen hier, nach Proxy suchen und sicherstellen, dass die Zeile für http_proxy & https_proxy leer ist.

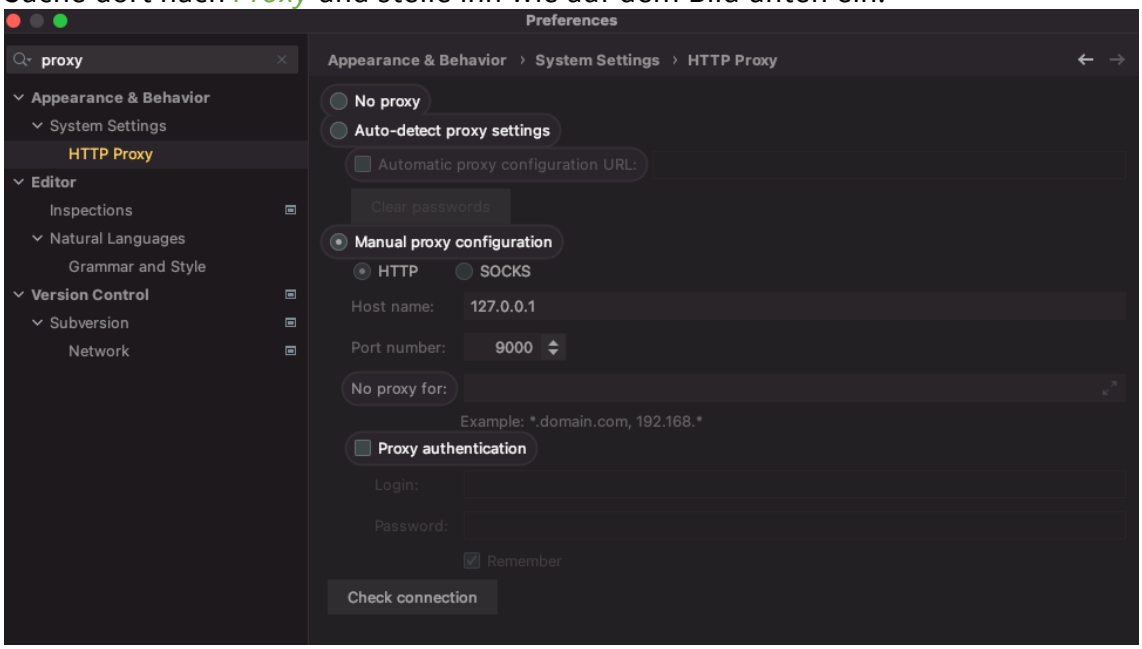


JetBrains (IntelliJ):

Statt Eclipse kannst du auch die IntelliJ IDE von JetBrains nutzen. Auch hier wieder: Proxy. Gehe also in die Einstellungen:



Suche dort nach *Proxy* und stelle ihn wie auf dem Bild unten ein:



Hinweis:

Auf diese Weise setzt du Proxy Einstellungen für ALLE JetBrains Programme. (PyCharm, DataGrip, usw...)

Für den Start in der Fakultät 73 solltest du aber unbedingt bei Eclipse bleiben. Eclipse ist zwar auch nicht meine erste Wahl, ist aber zum Coden lernen deutlich besser geeignet, weil:

- weniger automatisiert wird. Du musst viel selbst schreiben. Das trainiert. (Gerade Refactoring und Klassenstruktur)
- deine Dozenten kennen sich vermutlich deutlich besser mit Eclipse aus und können dir besser bei Problemen helfen.
- Die Oberfläche ist klarer strukturiert, du lernst leichter den Aufbau einer Java Applikation.

DBeaver:

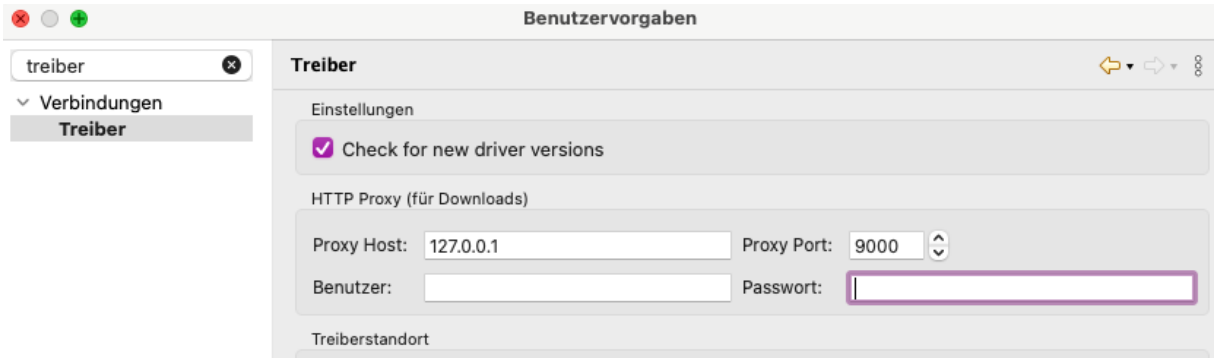
DBeaver ist ein Datenbank Management Tool. Installieren kannst du es über Homebrew:

```
brew install --cask dbeaver-community
```

Auch in DBeaver kann der Proxy eingestellt werden. Allerdings ist es durch den Zscaler nicht möglich, sich mit Datenbanken zu verbinden. Aus diesem Grund wirst du diese Software vermutlich niemals mit eingeschaltetem Zscaler verwenden und keine Einstellungen brauchen. Sollte es doch einmal notwendig sein, beachte, dass du den Proxy an zwei verschiedenen Orten konfigurieren musst:

Der erste Ort ist exakt wie bei Eclipse. Lese dort nach, wenn du nicht weißt, wie das geht.

Der zweite Ort dient zum Datenbank-Treiber Download. Öffne dazu wieder die *Preferences* und suche im Feld nach *Treiber*. Stelle dann die Proxy Adresse wie unten ein. Benutzer und Passwort sind nicht notwendig:

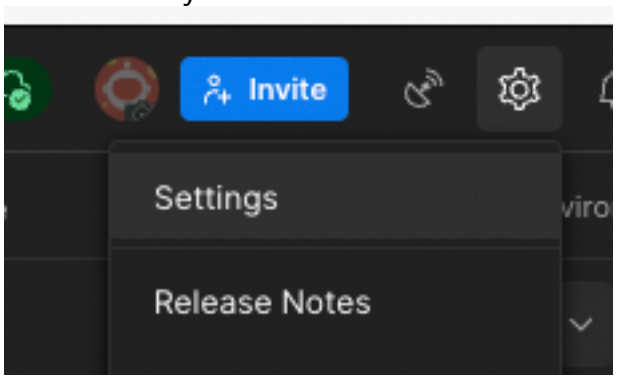


Postman:

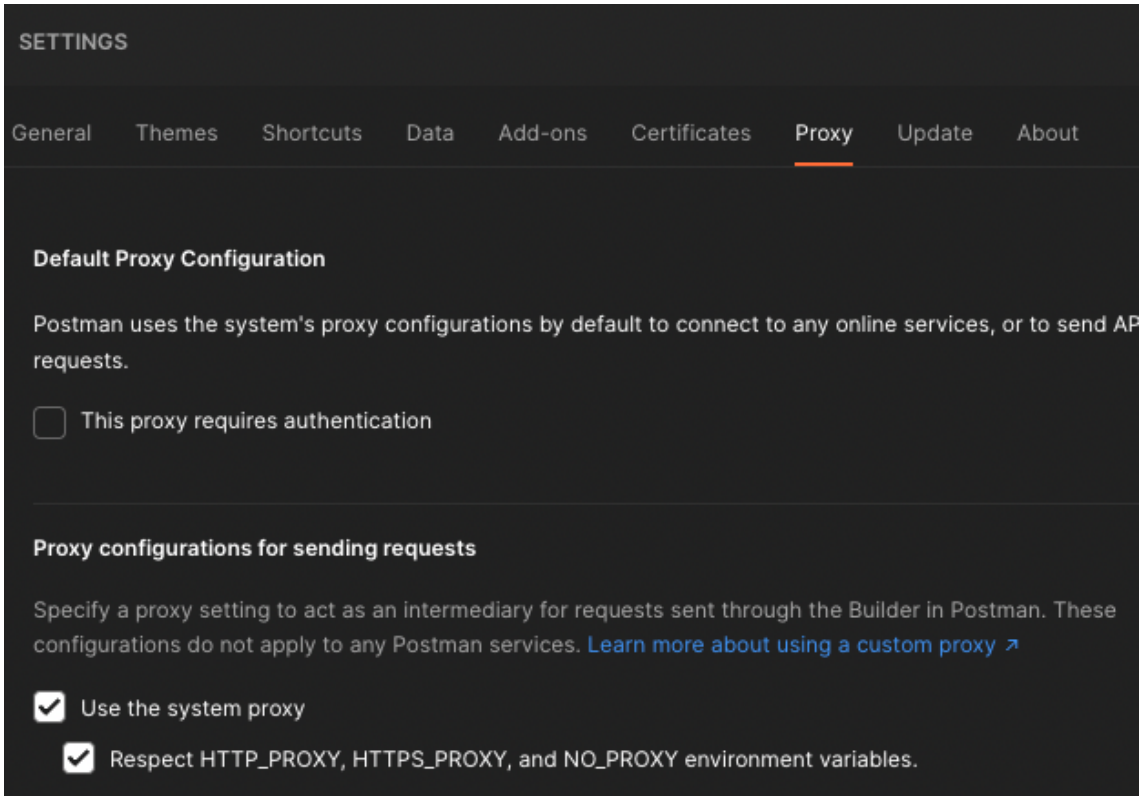
Postman ist ein Programm um REST Ressourcen zu konsumieren. Es ist extrem hilfreich zum Entwickeln von Web-Anwendungen. Auch Postman kommt via Homebrew:

```
brew install --cask postman
```

Und der Proxy:



Postman erkennt wie VS Code den Proxy aus der .zshrc-Datei automatisch, wenn die Haken wie auf dem Bild unten gesetzt werden.

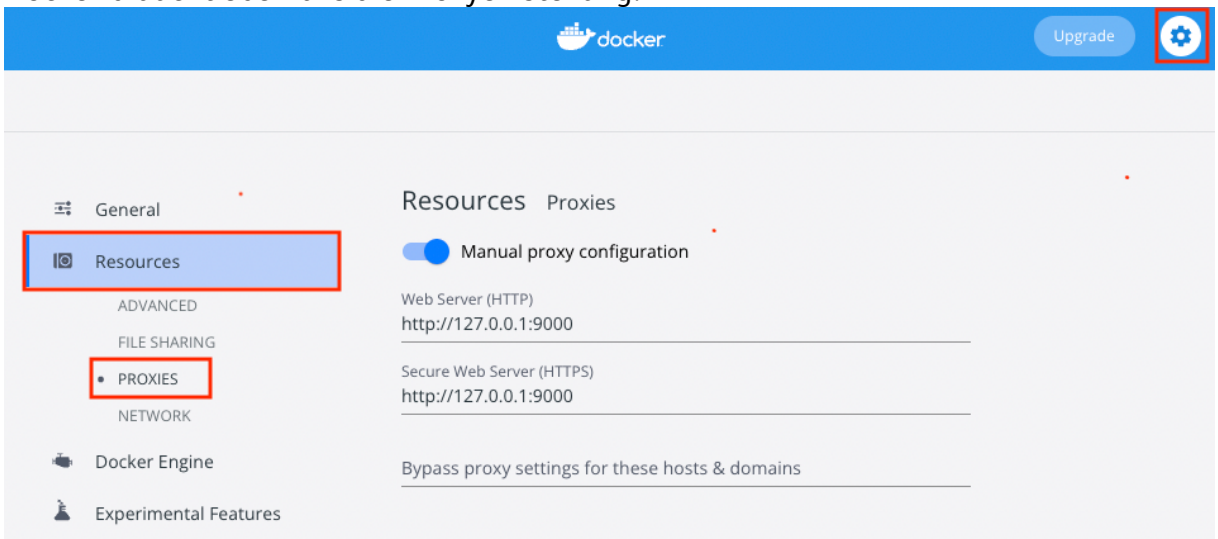


Docker:

Falls du Docker nutzten möchtest / musst, auch hier wieder: Homebrew.

```
brew install --cask docker
```

Docker braucht ebenfalls die Proxyeinstellung:



7. TIPPS & HINWEISE

Verstecke Dateien

Versteckte Dateien im Finder (zum Beispiel Dateien mit einem Punkt am Namensanfang) können mit folgenden Befehlen angezeigt werden:

```
defaults write com.apple.Finder AppleShowAllFiles true
killall Finder
```

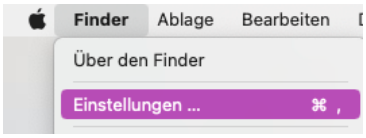
Um Dateien wieder auszublenden, den Wert einfach wieder auf false setzen und den Finder wieder neustarten.

```
defaults write com.apple.Finder AppleShowAllFiles false
killall Finder
```

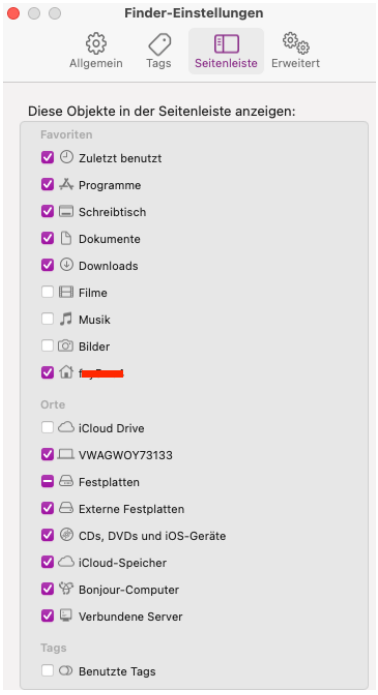
Finder optimieren

Im Finder wird standardmäßig nicht viel angezeigt. Als ehemaliger Windows Nutzer kann das sehr einschränkend sein. Um schneller mit dem Finder etwas zu finden, kann die Seitenleiste angepasst werden:

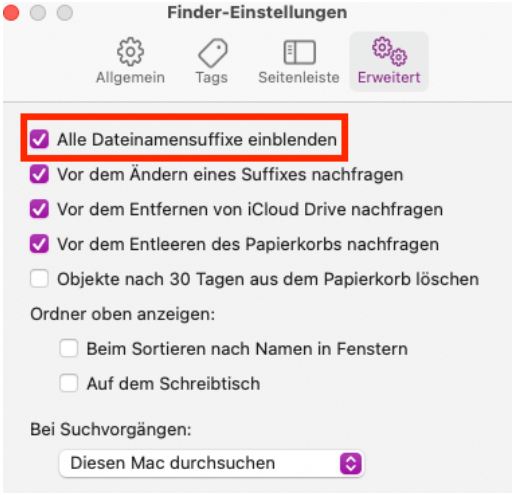
Zu den Einstellungen des Finders wechseln:



Seitenleiste anpassen:



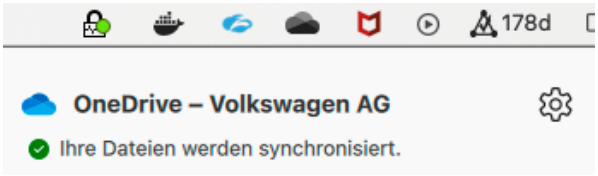
Erweiterte Einstellungen:



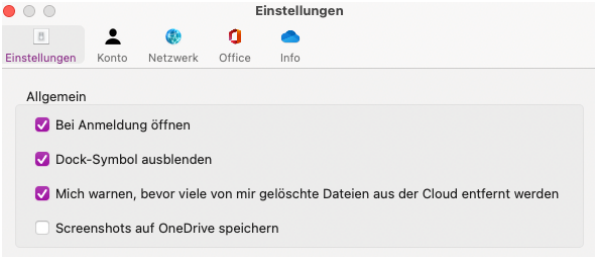
OneDrive synchron halten

Es kann passieren, dass plötzlich keine Dateien mehr aus dem Finder geöffnet werden können, die auf OneDrive liegen. Um Fehler vorzubeugen, sollte immer die Synchronisierung aktiviert sein

Das OneDrive Symbol sollte in der Statusleiste zu sehen sein. Bei einem Klick darauf, sollte es so aussehen wie auf dem Bild rechts.

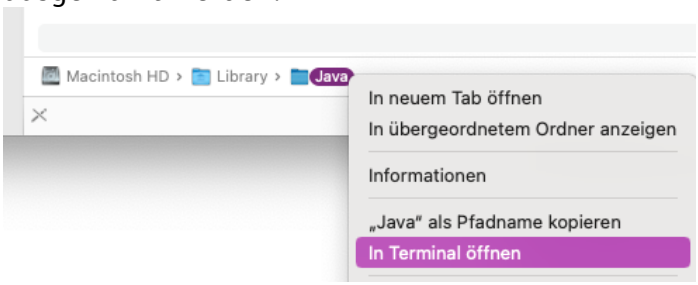


Um dafür zu sorgen, dass OneDrive immer Mit dem Mac startet, kann dies in den Einstellungen aktiviert werden.



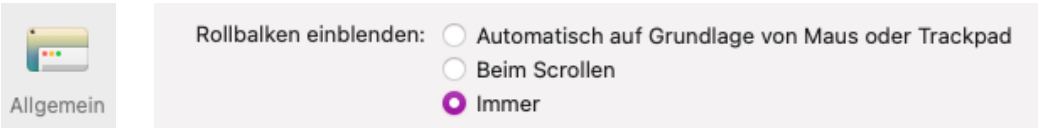
Pfade & Terminal Tipps

Um schneller an Pfadnamen zu kommen oder direkt ein Terminal auf einem Pfad zu öffnen, kann im Finder unten auf den Pfad rechts geklickt werden und die gewünschte ausgewählt werden:



Scrollbalken

MacOS baut alles auf die eigene Hardware auf, genauso auch auf die Magic Mouse. Per Standard blendet MacOS Scrollbalken aus und nur beim Darüberhovern wieder ein. Das hat mich vor allem beim Arbeiten mit Eclipse und IntelliJ gestört. Glücklicherweise kann man das Verhalten in den Systemeinstellungen ändern:



8. NACHWORT

Wie eingangs erwähnt, ist dies ein freiwilliges Werk. Es hat keinen offiziellen Bezug zu Volkswagen oder der Fakultät73, deswegen wird dort auch kein Support für diese Hilfestellungen angeboten.

Natürlich habe ich alle Einstellungen – vor allem das *install.sh*-Skript, ausgiebig getestet. Dennoch ist nicht auszuschließen, dass mal etwas schief geht. Als angehende Junior Software Entwickler wird es euer Job sein, Probleme zu lösen – auch solche, die banal mit einem PC zu tun haben. Schaut also zuallererst einmal selbst, ob ihr etwas Lösen könnt, bevor ihr mich kontaktiert.

Falls ihr dennoch nicht weiterkommt:
tobias.graetsch@volkswagen.de

9. QUELLEN

bell-sw/homebrew-liberica (github.com)

How to set or change the default Java (JDK) version on macOS? - Stack Overflow

Homebrew - Basics Commands and Cheatsheet - DEV Community

Homebrew/install: Homebrew (un)installer (github.com)