324-05B – Artefakte deployen

Einleitung

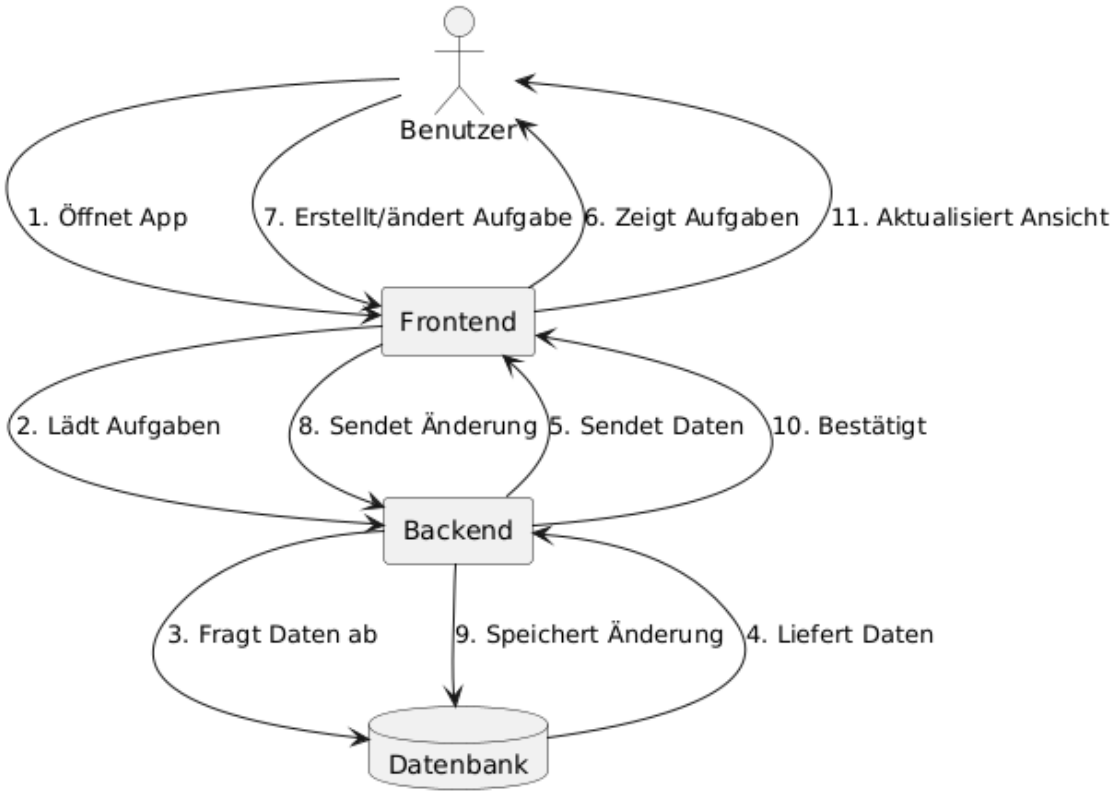
Wir arbeiten an dieser Version unserer To-Do-App:
https://gitlab.com/me7554020/m324_projekt_todolist-gemeinsam2/-/tree/ccce165d723d4e731e4fc8e4863742b0695e5af6

Es ist die Version nach dem unsere vier User-Stories und die Pipeline implementiert wurden.

Konzept wie mit der To-Do App gearbeitet werden soll

Die To-Do App bietet einen unkomplizierten Weg, Aufgaben zu verwalten. Benutzer können neue Aufgaben mit Beschreibung, Priorität und Fälligkeitsdatum erstellen. Die Liste zeigt alle Aufgaben an, wobei verschiedene Prioritäten farblich markiert werden (rot für hoch, schwarz für mittel, grün für niedrig).

Mit einem Klick auf die Checkbox kann eine Aufgabe als erledigt markiert werden. Nicht mehr benötigte Aufgaben lassen sich einfach löschen. Alle Daten werden in einer Datenbank gespeichert und bleiben auch nach einem Neustart erhalten.



To-Do-App deployen

System aktualisieren und Firewall/Ports aktivieren

Zuerst einmal wird das Ubuntu-System zur Sicherheiut aktualisiert und die Firewall/Ports aktiviert damit man später problemlos Zugriff auf den Web-Server hat.

```
kris@kris-VirtualBox:~$ sudo apt update && sudo apt upgrade -y
sudo ufw enable
sudo ufw allow ssh
sudo ufw allow 80
sudo ufw allow 8080
```

Apache Webserver installieren

Danach wir der Apache Webserver installiert worauf später das Frontend laufen wird und somit zugriffbar gemacht wird.

```
kris@kris-VirtualBox:~$ sudo apt install apache2 -y
sudo systemctl start apache2
sudo systemctl enable apache2
sudo a2enmod rewrite
sudo systemctl restart apache2
```

Java installieren

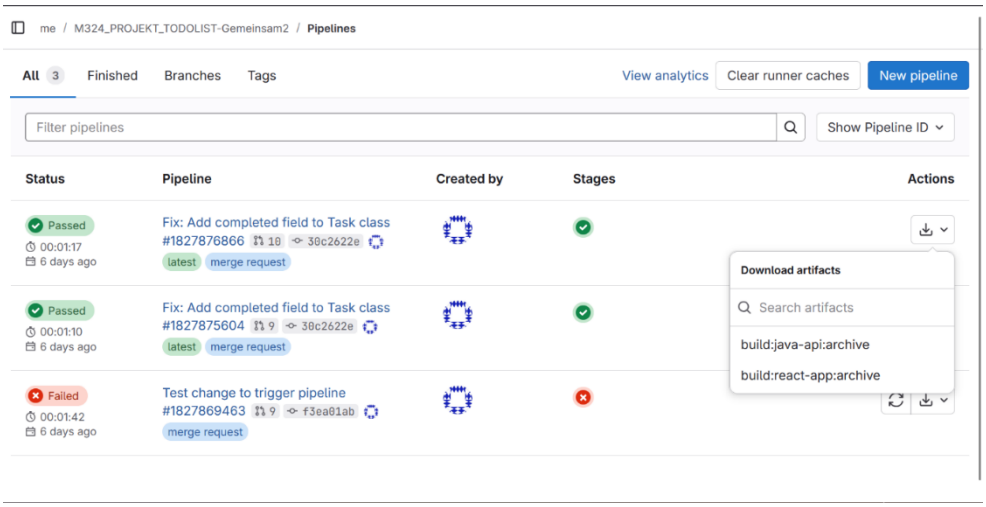
Java 17 war schon installiert und somit war es nicht notwendig es zu installieren.

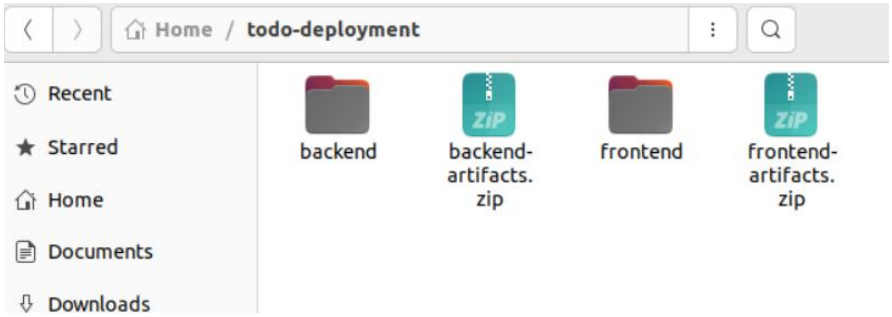
```
kris@kris-VirtualBox:~$ java -version
openjdk version "17.0.15" 2025-04-15
OpenJDK Runtime Environment (build 17.0.15+6-Ubuntu-0ubuntu122.04)
OpenJDK 64-Bit Server VM (build 17.0.15+6-Ubuntu-0ubuntu122.04, mixed mode, sharing)
```

Deployment vorbereiten

Artefakte downloaden

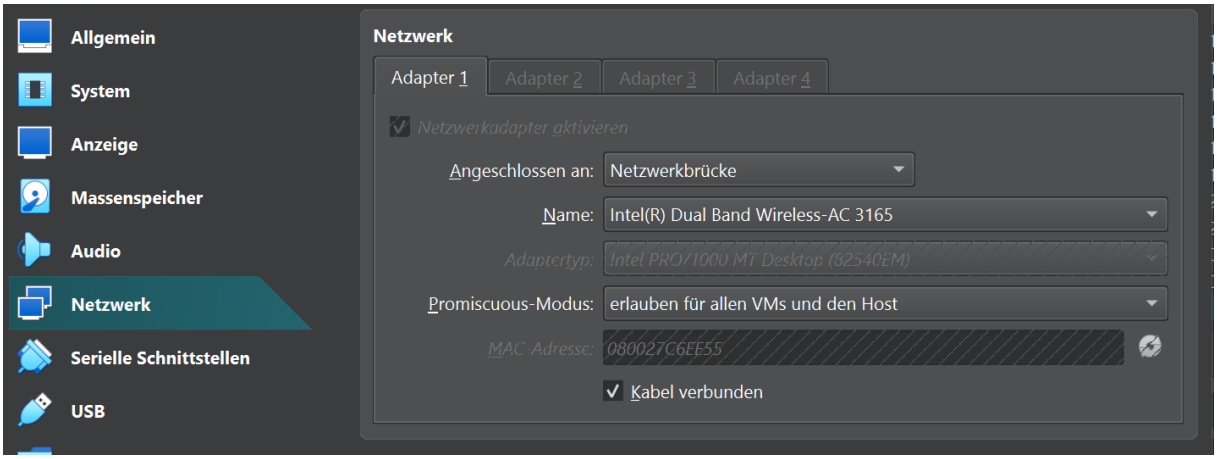
Die Artefakte wurden gedownloadet und dann in ein Verzeichnis gelegt spezifisch für diese SideQuest wo sie ebenfalls entpackt wurden.





IP-Addresse einrichten

Damit später vom Host her aus der Zugriff möglich ist musste die Netzwerkeinstellung der VM geändert werden in dem von NAT auf Netzwerkbrücke umgeschaltet wurde.



Frontend deployen

API-URLs anpassen

```
find frontend/build -name "*.js" -exec sed -i "s/localhost:8080/$SERVER_IP:8080/g" {} \;
```

Hardcodierte localhost-URLs werden durch die Server-IP ersetzt, damit das Frontend das Backend erreichen kann.

React Build zu Apache kopieren

Die React-Build-Dateien werden ins Apache-Webroot kopiert und mit korrekten Berechtigungen versehen.

```
sudo rm -rf /var/www/html/*

sudo cp -r frontend/build/* /var/www/html/

sudo chown -R www-data:www-data /var/www/html

sudo chmod -R 755 /var/www/html
```

Apache Virtual Host konfigurieren

```
sudo tee /etc/apache2/sites-available/todo-app.conf > /dev/null <<EOF
<VirtualHost *:80>
  DocumentRoot /var/www/html
  <Directory /var/www/html>
    Options -Indexes +FollowSymLinks
    AllowOverride All
    Require all granted
    RewriteEngine On
    RewriteBase /
    RewriteRule ^index\.html$ - [L]
    RewriteCond %{REQUEST_FILENAME} !-f
    RewriteCond %{REQUEST_FILENAME} !-d
    RewriteRule . /index.html [L]
  </Directory>
</VirtualHost>
EOF
```

```
sudo a2ensite todo-app.conf

sudo a2dissite 000-default.conf

sudo systemctl reload apache2
```

Apache wird für React Router konfiguriert, sodass alle unbekannten Routen zu index.html weiterleiten.

Backend deployen

Deployment-Verzeichnis vorbereiten

```
sudo mkdir -p /opt/todo-app

sudo cp backend/target/demo-0.0.1-SNAPSHOT.jar /opt/todo-app/

sudo useradd --system --shell /bin/false todo-app

sudo chown -R todo-app:todo-app /opt/todo-app
```

Die JAR-Datei wird nach /opt/todo-app kopiert und ein dedizierter Service-User erstellt.

Systemd Service konfigurieren

```
sudo tee /etc/systemd/system/todo-app.service > /dev/null <<EOF

[Unit]

Description=ToDo App Spring Boot Application

After=network.target

[Service]

Type=exec

User=todo-app

WorkingDirectory=/opt/todo-app

ExecStart=/usr/lib/jvm/java-17-openjdk-amd64/bin/java -jar /opt/todo-app/demo-0.0.1-SNAPSHOT.jar

Restart=on-failure

RestartSec=10

StandardOutput=journal

StandardError=journal

Environment=JAVA_HOME=/usr/lib/jvm/java-17-openjdk-amd64

Environment=SERVER_PORT=8080
```

```
[Install]

WantedBy=multi-user.target

EOF
```

Ein Systemd-Service wird erstellt, der das Backend automatisch startet und bei Fehlern neustartet.

Service starten

```
sudo systemctl daemon-reload

sudo systemctl enable todo-app.service

sudo systemctl start todo-app.service
```

Der Service wird aktiviert und gestartet, sodass er beim Boot automatisch läuft.

Resultat

Das Deployment war erfolgreich - die ToDo-App ist vollständig funktionsfähig und vom Host-Computer aus über die VirtualBox VM erreichbar. Der Screenshot zeigt die React-Anwendung unter http://192.168.1.129/, die korrekt vom Apache Webserver ausgeliefert wird.