

InnoLab-Mednanny

Projektdokumentation

Anforderungen.....	3
Planung und Durchführung.....	3
Projektbeschreibung.....	4
docker-compose.yml.....	5
Tomcat.....	7
Bestehende Probleme.....	7
BenutzerInnen Dokumentation.....	9
Docker.....	9
Adminer.....	9
Tomcat.....	10
Links.....	11

Anforderungen

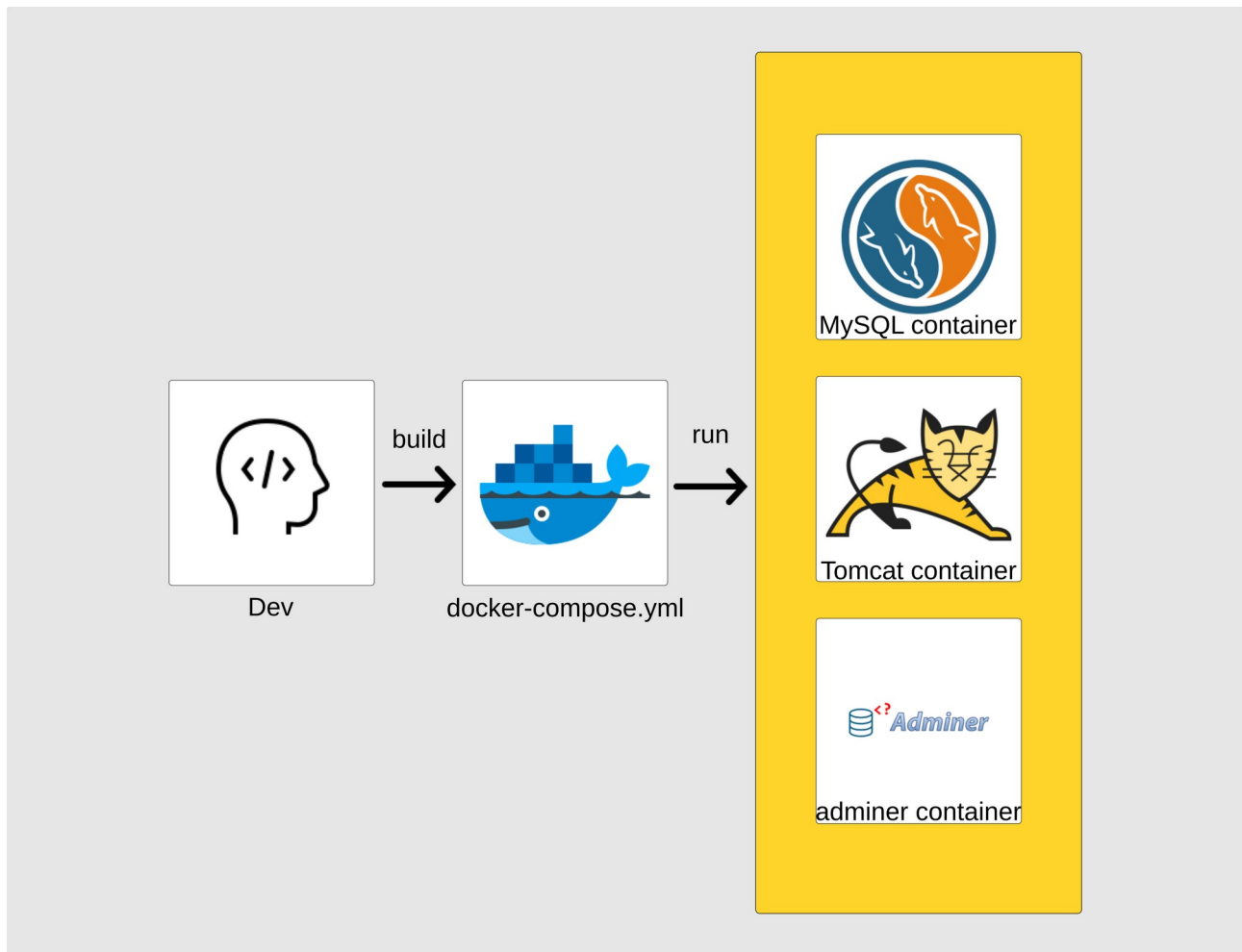
Ziel des Projektes war ein altes Server-Image mithilfe von Apache Tomcat und MySQL unter der Verwendung von Docker neu aufzusetzen. Durch die erstellten Dockercontainer soll die Plattform leicht auf einem Server bereitgestellt werden, um sie später zu testen. Dabei war grundsätzlich zu beachten, dass die mitgelieferten Datenbankskripts verlässlich ausgeführt und die Datenbankverbindung des Apache Tomcat-Servers zum MySQL Server funktioniert. Des Weiteren sollten alle statischen Seiten als auch alle Servlets des Tomcat Servers erreichbar sein.

Planung und Durchführung

Da das Server-Image schon bereitgestellt wurde und die zu verwendenden Technologien und das genaue Ziel schon vorab bekannt waren, war die Planung diesbezüglich zu vernachlässigen. Da sowohl die Verwendung von Docker als auch von Apache Tomcat für mich neu waren, war von Beginn an ein Großteil der Zeit für das Einlesen, Verstehen und Testen mit diesen Technologien vorgesehen.

Eine Teamplanung war nicht nötig, da das Projekt allein umgesetzt wurde. Einer strenger Zeitplan wann, was erledigt werden sollte, wurde nicht erstellt. Beim Arbeiten am Projekt wurde der als Nächstes umzusetzende Schritt meist ersichtlich. Grundsätzlich wurde ein GitHub Repository erstellt, um den Fortschritt zu speichern und zu dokumentieren. Als grundsätzliches Problem der Durchführung des Projekts stellten sich fehlende Informationen, Konfigurationen oder Dateien heraus, ohne die ein Weiterarbeiten nicht möglich war und somit bis zum Erhalt dieser den Fortschritt blockierten.

Projektbeschreibung



Im Allgemeinen beruht das Projekt auf drei Services: Apache Tomcat, MySQL und Adminer. Über Apache Tomcat kann die Website und ihre Services aufgerufen. MySQL stellt die Datenbank bereit, welche über Adminer verwaltet werden kann. Die Services laufen dabei jeweils in einem eigenen Container. Der Adminer Container ist prinzipiell optional, da er für das Funktionieren des Projektes nicht erforderlich ist, erleichtert aber den Zugang und die Verwaltung der Datenbank.

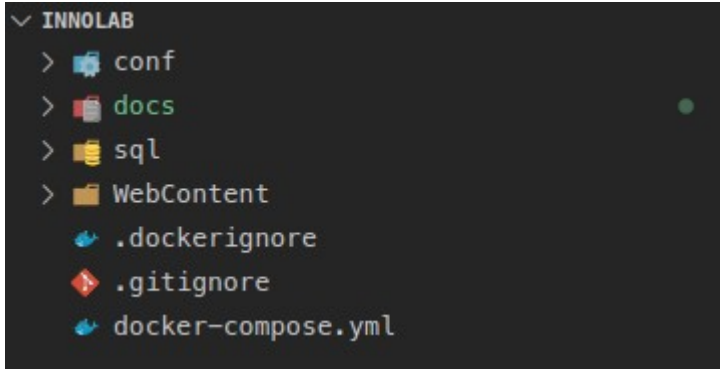
Als Basis für die Container wurden folgende auf Dockerhub verfügbaren Images verwendet:

- MySQL (Version 5.7): https://hub.docker.com/_/mysql
- Tomcat (Version 9.0.68): https://hub.docker.com/_/tomcat
- Adminer (latest): https://hub.docker.com/_/adminer/

Da für die vorweg schon bekannten Technologien bereits offizielle Docker Images existieren, wurden keine neuen Docker Images erstellt, sondern diese lediglich in einem docker-compose.yml File konfiguriert. Der Großteil der Arbeit wurde somit auch mit der Konfiguration dieses Files verbracht.

docker-compose.yml

Ordnerstruktur



Die Ordnerstruktur zur Verwendung der Applikation sollte wie auf dem Bild aussehen und kann unter <https://github.com/Teezious/inno-lab> geklont werden. (Der docs Ordner kann ignoriert werden). Jegliche Konfigurationen für Tomcat oder MySQL befinden sich im conf Ordner. Zusätzliche Konfigurationen können ebenfalls hinzugefügt werden, müssen aber entsprechend im docker-compose.yml angepasst werden. Die Datenbankskripts liegen im sql Ordner und können erweitert werden. Alle für die Webapplikation relevanten Files sollten in den WebContent Ordner platziert werden. (**WEB-INF** und **META-INF**)

Datenbank

```
db:
  container_name: mysql
  image: mysql:5.7
  restart: "always"
  command: --default-authentication-plugin=mysql_native_password
  environment:
    MYSQL_ROOT_PASSWORD: mednannypw
    MYSQL_DATABASE: mednanny
    MYSQL_USER: testuser
    MYSQL_PASSWORD: mednannypw
  ports:
    - "3306:3306"
  volumes:
    - ./sql:/docker-entrypoint-initdb.d
    - /etc/localtime:/etc/localtime:ro
    - ./conf/my.cnf:/etc/my.cnf
```

Als MySQL Version wird unter dem Punkt image definiert, dass **Version 5.7** verwendet werden soll. Hier wird 5.7 verwendet, da Adminer nur mit dieser Version funktionieren zu scheint. Restart always bedeutet, dass der Container immer neu gestartet wird, wenn er stoppt. Sollte der Container manuell gestoppt werden, startet er wieder neu, sofern der Docker daemon restartet oder der Container manuell wieder gestartet wird.

„--default-authentication-plugin=mysql_native_password„ unter commmand legt die

Authentifizierungsmethode fest.

Unter environment wird das root password, die Database und die ein testuser definiert. Das Passwort für den root Nutzer ist verpflichtend.

Ports definiert die Ports für den MySQL Container innerhalb und außerhalb der Host Machine. In diesem Fall für beide 3306: (Weiter Informationen <https://docs.docker.com/compose/networking/>) Unter dem Punkt Volumes können Dateien oder Ordner in den Container gemounted werden. Hier ist vor „./sql:/docker-entrypoint-initdb.d“ relevant. Im sql Ordner auf unserem Dateiensystem werden alle Skripte platziert die zum Start des Containers ausgeführt werden sollen. Diese werden dann im Container gemounted und zum Start ausgeführt. Unter „./conf/my.cnf“ kann die generelle Konfiguration für den MySQL Server definiert werden.

Tomcat

```
tomcat:
  container_name: tomcat
  image: tomcat:9.0.68
  ports:
    - 8080:8080
  volumes:
    - ./WebContent:/usr/local/tomcat/webapps/mednanny/
    - /etc/localtime:/etc/localtime:ro
    - ./conf/context.xml:/usr/local/tomcat/conf/context.xml
    - ./conf/server.xml:/usr/local/tomcat/conf/server.xml
  depends_on:
    - db
  environment:
    JDBC_URL: jdbc:mysql://db:3306/mednanny?useUnicode=true&characterEncoding=UTF-8&autoReconnect=true&failOverReadOnly=false&maxReconnects=50
    JDBC_ROOTUSER: root
    JDBC_PASS: mednannypw
  restart: "always"
```

Der docker-compose.yml Teil für Tomcat ist recht ähnlich wie der von MySQL. Beim Image wurde hier die **Version „9.0.68“** verwendet, da es die letzte ist für die, die Servlets des Serverimage funktionieren.

Unter environment werden die Environment-Variablen definiert die später im Container verwendet werden können.

Unter volumes werden die verschiedenen Config-Files in den Tomcat Container gemounted.

Wichtig ist ebenfalls das mounten des WebContent Ordners in den „webapps/mednanny“ Ordner des Tomcat Containers.

Depends_on definiert die Startreihenfolge des Containers. In diesem Fall soll Tomcat nach der Datenbank gestartet werden. Tomcat kann unter <http://localhost:8080/mednanny/> + “servlet-url“ verwendet werden

Adminer

```
adminer:
  container_name: adminer
  image: adminer:latest
  restart: "always"
  ports:
    - 8081:8080
  environment:
    ADMINER_DESIGN: galkaev
    ADMINER_DEFAULT_SERVER: db
  depends_on:
    - db
  volumes:
    - /etc/localtime:/etc/localtime:ro
```

Der Adminer Teil ist ebenfalls ident zu den vorherigen zwei Abschnitten. Adminer kann unter <http://localhost:8081/> verwendet werden.

Tomcat

WebContent web.xml

Unter WEB-INF/web.xml werden alle servlets und die URL, mit der sie aufgerufen werden, definiert.

WebContent applicationContext.xml

Unter WEB-INF/applicationContext.xml wird definiert welche Datenbankverbindung und welcher DAO User verwendet werden. Sowohl die Datenbankverbindung als auch der DAO User müssen vorher definiert sein. In diesem Projekt wurde der DAO-User in der XML Datei vorerst auskommentiert, da die fehlende Konfiguration zu Fehlern beim Ausführen von Tomcat führt.

WebContent context.xml

Unter META-INF/context.xml wird die zu verwendenden Datasources, die dann im applicationContext.xml verwendet definiert. Auch der DAO User sollte hier definiert werden. Die Datasource wurde für dieses Projekt definiert, allerdings scheint die Datenverbindung zum MySQL Server nicht zu funktionieren. Der Grund hierfür konnte leider nicht gefunden werden.

conf context.xml

Hier können die Resources definiert werden, deren Änderung ein Reload der Applikation hervorrufen soll. (gilt für den ganzen Server). Außerdem wurde hier Caching eingeschalten und die Größe festgelegt.

conf server.xml

Hier können ebenfalls Einstellungen für den ganzen Server festgelegt werden. Etwa eine Datenbankverbindung oder das Logging. Für dieses Projekt blieb server.xml aber unverändert.

Bestehende Probleme

```
tomcat | com.mysql.jdbc.exceptions.jdbc4.CommunicationsException: Communications link failure
tomcat |
tomcat | The last packet sent successfully to the server was 0 milliseconds ago. The driver has not received any packets from the server.
tomcat | Communications link failure
tomcat |
tomcat | The last packet sent successfully to the server was 0 milliseconds ago. The driver has not received any packets from the server.
tomcat | com.mysql.jdbc.exceptions.jdbc4.CommunicationsException: Communications link failure
tomcat |
tomcat | The last packet sent successfully to the server was 0 milliseconds ago. The driver has not received any packets from the server.com.mysql.jdbc.exceptions.jdbc4.CommunicationsException: Communications link failure
tomcat |
tomcat | The last packet sent successfully to the server was 0 milliseconds ago. The driver has not received any packets from the server.
tomcat | Communications link failure
tomcat |
tomcat | The last packet sent successfully to the server was 0 milliseconds ago. The driver has not received any packets from the server.
tomcat | com.mysql.jdbc.exceptions.jdbc4.CommunicationsException: Communications link failure
tomcat |
tomcat | The last packet sent successfully to the server was 0 milliseconds ago. The driver has not received any packets from the server.
```

Leider kann keine Datenbankverbindung hergestellt werden. Das Problem konnte, trotz vieler verschiedener Lösungsansätze nicht gelöst werden.

Installiert man in der Tomcat Shell einen MySQL Client, ist eine Verbindung zum anderen Container aber trotzdem möglich wie im folgenden Screenshot ersichtlich ist. Das Problem könnte als an der App selbst liegen.

```

docker-compose.yml M applicationContext.xml X
WebContent > WEB-INF > applicationContext.xml > _

1 <?xml version="1.0" encoding="UTF-8"?>
2 <beans xmlns="http://www.springframework.org/schema/beans"
3     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4     xmlns:p="http://www.springframework.org/schema/p"
5     xmlns:aop="http://www.springframework.org/schema/aop"
6     xmlns:tx="http://www.springframework.org/schema/tx"
7     xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
8     http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-2.5.xs
9     http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-2.5.xsd">
10
11
12 <bean id="dataSource" class="org.springframework.jndi.JndiObjectFactoryBean">
13     <property name="jndiName" value="java:comp/env/jdbc/mednanny" />
14 </bean>
15
16 <!-- <bean id="UserDao" class="Dao.UserDao">
17     <constructor-arg ref="dataSource"/>
18 </bean -->
19
20 </beans>
21

```

Des Weiteren besteht das Problem, dass wenn folgende obige kommentierten Zeilen wieder auskommentiert werden der folgende Error geworfen wird.

```

root@9133ec664b:/usr/local/tomcat# cat logs/localhost.2022-11-01.log
01-Nov-2022 21:57:13.994 INFO [main] org.apache.catalina.core.ApplicationContext.log Initializing Spring root WebApplicationContext
01-Nov-2022 21:57:14.154 SEVERE [main] org.apache.catalina.core.StandardContext.listenerStart Exception sending context initialized event to listener instance of class [org.springframework.web.context.ContextLoaderListener]
org.springframework.beans.factory.CannotLoadBeanClassException: Cannot find class [Dao.UserDao] for bean with name 'UserDao' defined in ServletContext resource [/WEB-INF/applicationContext.xml]; nested exception is java.lang.ClassNotFoundException:
Dao.UserDao
    at org.springframework.beans.factory.support.AbstractBeanFactory.resolveBeanClass(AbstractBeanFactory.java:1141)
    at org.springframework.beans.factory.support.AbstractBeanFactory.predictBeanType(AbstractBeanFactory.java:524)
    at org.springframework.beans.factory.support.AbstractBeanFactory.isFactoryBean(AbstractBeanFactory.java:1177)
    at org.springframework.beans.factory.support.AbstractBeanFactory.isFactoryBean(AbstractBeanFactory.java:758)
    at org.springframework.beans.factory.support.DefaultListableBeanFactory.preInstantiateSingletons(DefaultListableBeanFactory.java:422)
    at org.springframework.context.support.AbstractApplicationContext.refresh(AbstractApplicationContext.java:728)
    at org.springframework.context.support.AbstractApplicationContext.refresh(AbstractApplicationContext.java:380)
    at org.springframework.web.context.ContextLoader.createWebApplicationContext(ContextLoader.java:255)
    at org.springframework.web.context.ContextLoader.initWebApplicationContext(ContextLoader.java:199)
    at org.springframework.web.context.ContextLoaderListener.contextInitialized(ContextLoaderListener.java:45)
    at org.apache.catalina.core.StandardContext.listenerStart(StandardContext.java:4769)
    at org.apache.catalina.core.StandardContext.startInternal(StandardContext.java:5231)
    at org.apache.catalina.util.LifecycleBase.start(LifecycleBase.java:183)
    at org.apache.catalina.core.ContainerBase.addChildInternal(ContainerBase.java:726)
    at org.apache.catalina.core.ContainerBase.addChild(ContainerBase.java:698)
    at org.apache.catalina.core.StandardHost.addChild(StandardHost.java:696)
    at org.apache.catalina.startup.HostConfig.deployDirectory(HostConfig.java:1185)
    at org.apache.catalina.startup.HostConfig.deployApps(HostConfig.java:1933)
    at java.base/java.util.concurrent.Executors$RunnableAdapter.call(Executors.java:539)
    at java.base/java.util.concurrent.FutureTask.run(FutureTask.java:264)
    at org.apache.tomcat.util.threads.InlineExecutorService.execute(InlineExecutorService.java:75)
    at java.base/java.util.concurrent.AbstractExecutorService.submit(AbstractExecutorService.java:123)
    at org.apache.catalina.startup.HostConfig.deployDirectories(HostConfig.java:1095)
    at org.apache.catalina.startup.HostConfig.deployApps(HostConfig.java:477)
    at org.apache.catalina.startup.HostConfig.start(HostConfig.java:1618)
    at org.apache.catalina.util.LifecycleBase.fireLifecycleEvent(LifecycleBase.java:122)
    at org.apache.catalina.util.LifecycleBase.setStateInternal(LifecycleBase.java:423)
    at org.apache.catalina.util.LifecycleBase.setState(LifecycleBase.java:366)
    at org.apache.catalina.core.ContainerBase.startInternal(ContainerBase.java:946)
    at org.apache.catalina.core.StandardHost.startInternal(StandardHost.java:935)
    at org.apache.catalina.util.LifecycleBase.start(LifecycleBase.java:183)
    at org.apache.catalina.core.ContainerBase.startChild.call(ContainerBase.java:1396)
    at org.apache.catalina.core.ContainerBase.startChild.call(ContainerBase.java:1386)
    at java.base/java.util.concurrent.FutureTask.run(FutureTask.java:264)
    at org.apache.tomcat.util.threads.InlineExecutorService.execute(InlineExecutorService.java:75)
    at java.base/java.util.concurrent.AbstractExecutorService.submit(AbstractExecutorService.java:145)
    at org.apache.catalina.core.ContainerBase.startInternal(ContainerBase.java:919)
    at org.apache.catalina.core.StandardEngine.startInternal(StandardEngine.java:265)
    at org.apache.catalina.util.LifecycleBase.start(LifecycleBase.java:183)
    at org.apache.catalina.core.StandardService.startInternal(StandardService.java:432)
    at org.apache.catalina.util.LifecycleBase.start(LifecycleBase.java:183)
    at org.apache.catalina.core.StandardServer.startInternal(StandardServer.java:930)
    at org.apache.catalina.util.LifecycleBase.start(LifecycleBase.java:183)
    at org.apache.catalina.startup.Catalina.start(Catalina.java:772)
    at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke(Native Method)
    at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:77)
    at java.base/jdk.internal.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
    at java.base/java.lang.reflect.Method.invoke(Method.java:568)
    at org.apache.catalina.startup.Bootstrap.start(Bootstrap.java:345)
    at org.apache.catalina.startup.Bootstrap.main(Bootstrap.java:476)
Caused by: java.lang.ClassNotFoundException: Dao.UserDao
    at org.apache.catalina.loader.WebappClassLoaderBase.loadClass(WebappClassLoaderBase.java:1412)
    at org.apache.catalina.loader.WebappClassLoaderBase.loadClass(WebappClassLoaderBase.java:1220)
    at org.springframework.util.ClassUtils.forName(ClassUtils.java:211)
    at org.springframework.beans.factory.support.AbstractBeanDefinition.resolveBeanClass(AbstractBeanDefinition.java:385)
    at org.springframework.beans.factory.support.AbstractBeanFactory.resolveBeanClass(AbstractBeanFactory.java:1138)
    ... 58 more
01-Nov-2022 21:57:14.168 INFO [main] org.apache.catalina.core.ApplicationContext.log Closing Spring root WebApplicationContext
root@9133ec664b:/usr/local/tomcat#

```

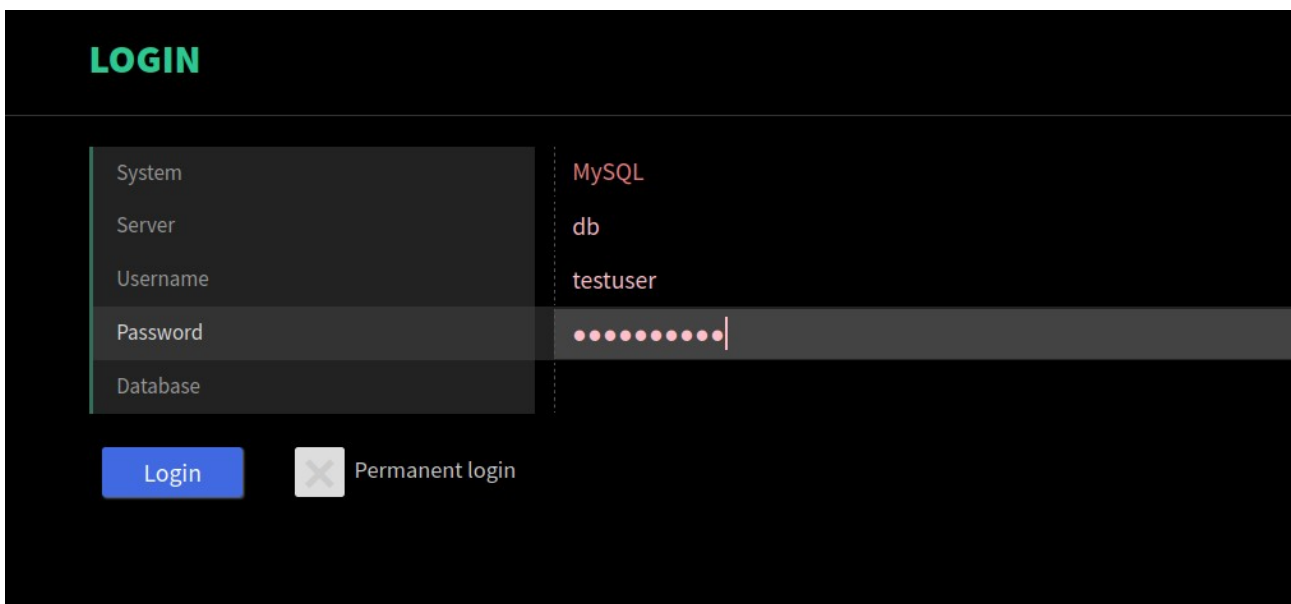

BenutzerInnen Dokumentation

Docker

- „docker-compose up -d“ um die Services zu builden und zu starten
- „docker-compose down“ um die Services zu beenden
- „docker-compose logs -f“ um die logs der Services einzusehen
- „docker system prune -a“ um alle images, volumes und networks zu löschen
- „docker ps -a“ um die laufenden container einzusehen
- „docker image ls“ um alle Images einzusehen
- „docker container ls“ um alle container einzusehen
- „docker exec -it <container-name> bash“ um eine shell zu attachen
- „docker inspect <container-name>“ um alle container informationen (etwa IP zu bekommen)

Adminer

Kann über <http://localhost:8081/> verwendet werden. Username und Password wie im docker-compose.yml File definiert.



The image shows the Adminer LOGIN interface. It has a dark background with a light green 'LOGIN' header. On the left, there is a form with five input fields: 'System', 'Server', 'Username', 'Password', and 'Database'. The 'Password' field is currently filled with red dots. On the right, the values entered are displayed: 'MySQL' for System, 'db' for Server, 'testuser' for Username, and a redacted password for Password. At the bottom, there is a blue 'Login' button and a checkbox labeled 'Permanent login' which is currently unchecked.

Adminer 4.8.1

DB: mednanny

SQL COMMAND IMPORT EXPORT CREATE TABLE

Tables and views

Search data in tables (230)

Table	Engine	Collation	Data Length	Index Length	Data Free	Auto Increment	Rows	Comment
a3admins	MyISAM	latin1_swedish_ci	0	1,024	0		48	0
ACAPW	MyISAM	latin1_swedish_ci	0	1,024	0		0	0
adminhistory	InnoDB	latin1_swedish_ci	16,384	0	0		1	0
aek_admins	InnoDB	latin1_swedish_ci	16,384	0	0		1	0
aek_aerzte	InnoDB	latin1_swedish_ci	16,384	0	0		1	0
aek_ext_arzt	InnoDB	latin1_swedish_ci	16,384	0	0		0	0
aek_zeit	InnoDB	latin1_swedish_ci	16,384	0	0		1	0
aek_zeit_kat	InnoDB	latin1_swedish_ci	16,384	0	0		1	0
ALALL	InnoDB	latin1_swedish_ci	16,384	0	0		1	0
ALLDANGER	MyISAM	latin1_swedish_ci	0	1,024	0		4	0
ALLREACT	MyISAM	latin1_swedish_ci	0	1,024	0		5	0
Apotheken	InnoDB	latin1_swedish_ci	16,384	0	0		1	0
arzt	MyISAM	latin1_swedish_ci	0	1,024	0	612,229	0	0
ARZT_FACHGEBIET	MyISAM	latin1_swedish_ci	0	1,024	0		0	0
Arzt_Smtp	MyISAM	latin1_swedish_ci	0	1,024	0		0	0
auspraegung	MyISAM	latin1_swedish_ci	0	1,024	0		5	0
BBRZ_AG	InnoDB	latin1_swedish_ci	16,384	0	0		1	0
BBRZ_DATUM	InnoDB	latin1_swedish_ci	16,384	0	0		0	0
BBRZ_TERMINE	InnoDB	latin1_swedish_ci	16,384	0	0		0	0

Tomcat

Kann über <http://localhost:8080/mednanny/> + <servlet-url> verwendet werden. Username und Password wie im docker-compose.yml File definiert.

mednanny [Arztuche](#) [Presse](#) [Über mednanny](#) [Kundenmeinungen](#) [LOGIN](#) [FÜR ÄRZTE](#) [DE](#)

3 Millionen Onlinebuchungen/Jahr

Ersparen Sie sich das Wartezimmer - Vereinbaren Sie Ihre Arzttermine online - Kostenlos und völlig sicher!

[Schnellsuche](#) [Erweiterte Suche](#)

Name Bundesland Fachrichtung Krankenkasse [SUCHEN & BUCHEN](#)

Aktuelle Schwerpunkte

Univ.-Prof. Dr. Johannes Huber informiert

Univ.-Prof. Dr. Norbert Bachl informiert

Univ.-Prof. Dr. Norbert Bachl informiert

Quickchecks

Nehmen Sie sich kurz Zeit und prüfen Sie Ihre Gesundheit und Erkrankungsrisiko ganz einfach mit unseren Online-Checks zu folgenden Themen:

- [Brustkrebsrisiko](#)
- [Prostata Symptomatik](#)

Links

Github: <https://github.com/Teezious/inno-lab/>

WebContent: <https://cloud.technikum-wien.at/s/r5A3GD7EGszC7Za>