

Курсов проект: Curdis

19113, 19117, 19104, 19125.

“Curdis” e online chat app

За направата на Curdis сме използвали Ubuntu:

Ubuntu е популярна операционна система базирана на Linux, тя е безплатна и с отворен код, което означава, че всеки може да я изтегли, да я инсталира и да я променя според своите нужди. Ubuntu е изградена върху основата на Debian Linux и се фокусира върху лесното използване, стабилността и сигурността.



LAMP

LAMP описва популярна стек от софтуерни компоненти, използвани за създаване на уебсайтове и уеб приложения. включва Linux OS, Apache HTTP сървър, MySQL база данни и PHP.

Linux

Linux е операционна система с отворен код, базирана на UNIX. Той е основата на LAMP стека и предоставя стабилност, сигурност и мощност за уеб приложения. Linux предлага голяма гъвкавост и множество инструменти за конфигурация и управление на сървърната инфраструктура.

Apache

Apache е най-популярният уеб сървър в света. Той предоставя услуги за доставка на уеб страници и уеб приложения към клиентските браузъри. Apache е известен със своята стабилност, сигурност и гъвкавост. С помощта на Apache можете да конфигурирате виртуални хостове, да управлявате URL пренасочвания и да използвате разширения и модули за различни функционалности.

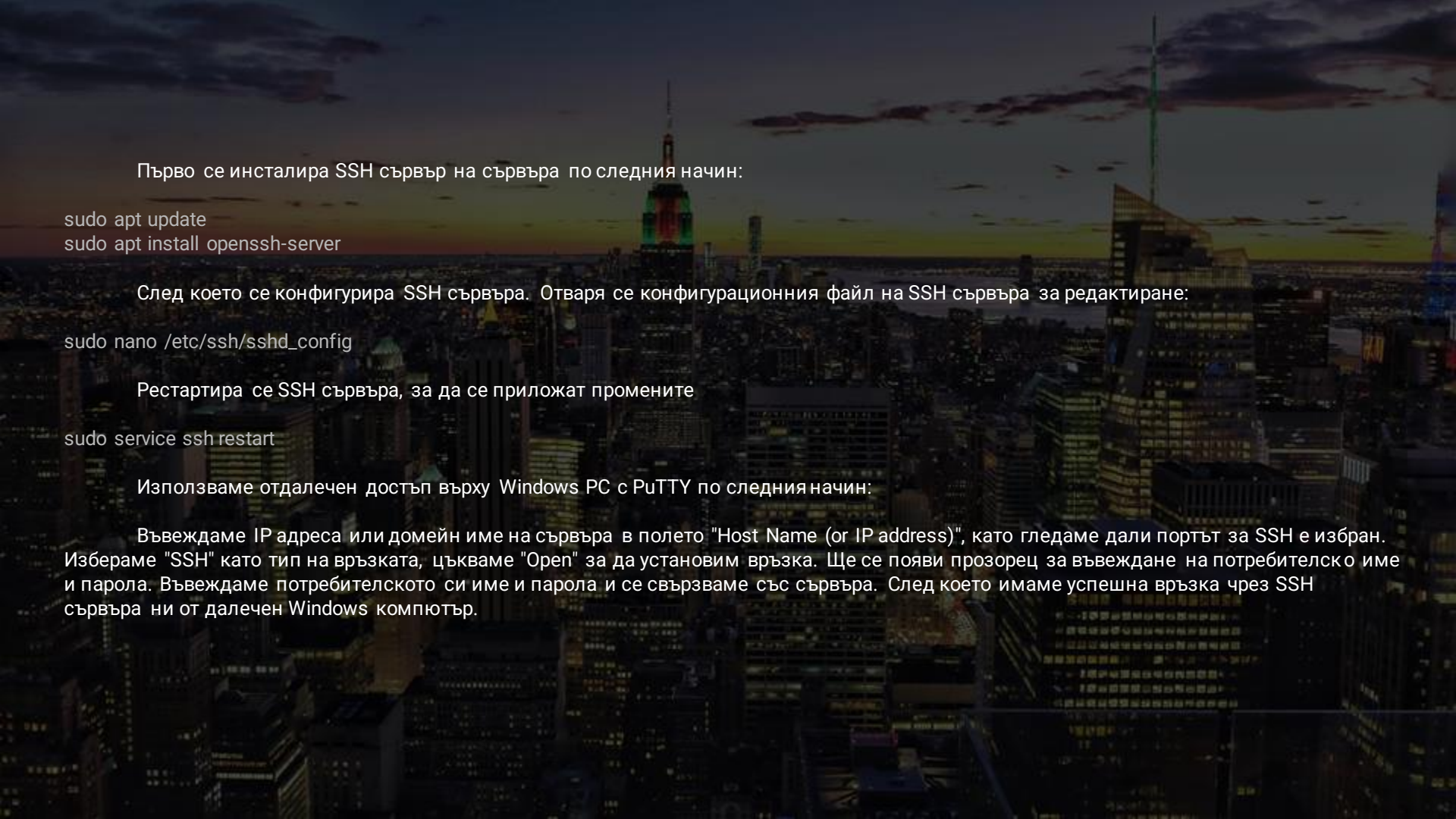
MySQL

MySQL е релационна база данни, която предоставя средства за съхранение, управление и извличане на данни. Те са много популярни в уеб разработката и осигуряват надеждна и ефективна система за управление на данни за уеб приложения. MySQL и поддържа SQL езика за работа с базата данни.

PHP

PHP е език за уеб разработка, който позволява създаването на динамични и интерактивни уеб страници. PHP има широко разпространение за разработка на уеб приложения.

Имаме файлов сървър vsftpd който предоставя отдалечено сваляне и качване на файлове през приложението FileZilla



Първо се инсталира SSH сървър на сървъра по следния начин:

```
sudo apt update  
sudo apt install openssh-server
```

След което се конфигурира SSH сървъра. Отваря се конфигурационния файл на SSH сървъра за редактиране:

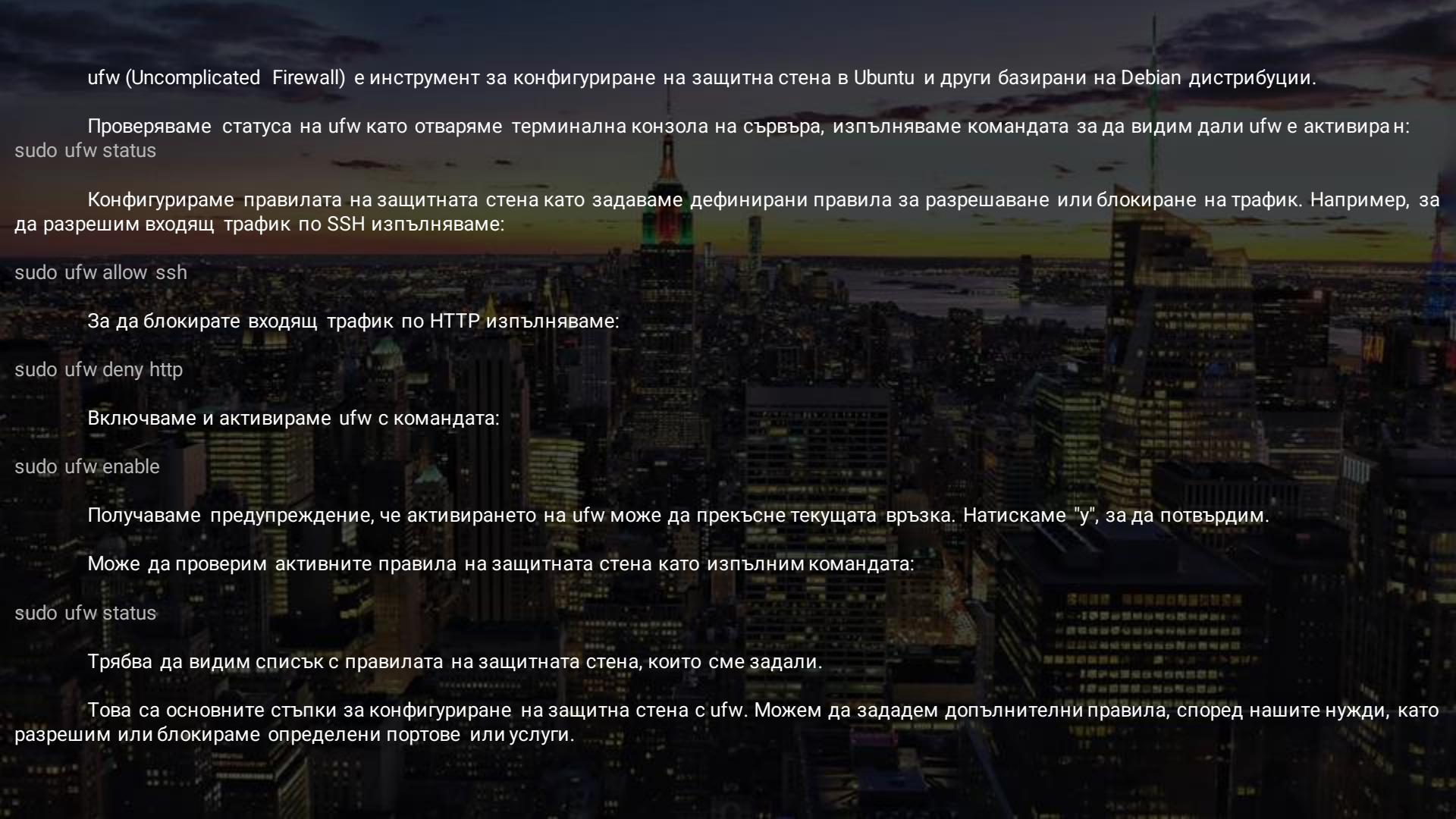
```
sudo nano /etc/ssh/sshd_config
```

Рестартира се SSH сървъра, за да се приложат промените

```
sudo service ssh restart
```

Използваме отдалечен достъп върху Windows PC с PuTTY по следния начин:

Въвеждаме IP адреса или домейн име на сървъра в полето "Host Name (or IP address)", като гледаме дали портът за SSH е избран. Избираме "SSH" като тип на връзката, цъкваме "Open" за да установим връзка. Ще се появи прозорец за въвеждане на потребителско име и парола. Въвеждаме потребителското си име и парола и се свързваме със сървъра. След което имаме успешна връзка чрез SSH сървъра ни от далечен Windows компютър.



ufw (Uncomplicated Firewall) е инструмент за конфигуриране на защитна стена в Ubuntu и други базирани на Debian дистрибуции.

Проверяваме статуса на ufw като отваряме терминална конзола на сървъра, изпълняваме командата за да видим дали ufw е активиран:

```
sudo ufw status
```

Конфигурираме правилата на защитната стена като задаваме дефинирани правила за разрешаване или блокиране на трафик. Например, за да разрешим входящ трафик по SSH изпълняваме:

```
sudo ufw allow ssh
```

За да блокирате входящ трафик по HTTP изпълняваме:

```
sudo ufw deny http
```

Включваме и активираме ufw с командата:

```
sudo ufw enable
```

Получаваме предупреждение, че активирането на ufw може да прекъсне текущата връзка. Натискам "y", за да потвърдим.

Може да проверим активните правила на защитната стена като изпълним командата:

```
sudo ufw status
```

Трябва да видим списък с правилата на защитната стена, които сме задали.

Това са основните стъпки за конфигуриране на защитна стена с ufw. Можем да зададем допълнителни правила, според нашите нужди, като разрешим или блокираме определени портове или услуги.

Създаване на автоматизиран Backup сценарий с Bash скрипт и CRON:

`nano backup_script.sh`

Въвеждаме следния код:

```
#!/bin/bash
SOURCE_DIR="/path/to/source"
BACKUP_DIR="/path/to/backup"
DATE=$(date +%Y%m%d%H%M%S)
mkdir -p $BACKUP_DIR/$DATE
cp -R $SOURCE_DIR/* $BACKUP_DIR/$DATE
echo "Backup created: $BACKUP_DIR/$DATE"
```

Добавяме изпълними права на скрипта

`chmod +x backup_script.sh`

Отваряме CRON таблицата за редактиране

`crontab -e`

Добавяме нов ред за да настроим колко често да backup-ва. Например всеки ден в 3 през нощта

`0 3 * * * /path/to/backup_script.sh`

За дизайна на Curdis

Исползвани технологии

Frontend: HTML5, CSS, JavaScript

Backend: PHP, MySQL

HTML

- Има три страници и 4 рорир прозореца, които служат в полза на потребителя. Рорир-овете са:
- *Friend Requests*
- *Color Theme Change*
- *ADD FRIENDS*
- *Profile picture or username change*
- За иконите са използвани "*IonIcons*"
- Чрез div-ове е разделен на различни контейнери.

CSS

- За цветовете на заден план е използван " *Background: linear-gradient ()*", като те могат да се сменят от настройките.
- Font Family – ' Popins ' от *Google Fonts*
- Главната страница е разделена на две части - лявата е за настройки и избиране на чат. Дясната е за изпращане на съобщения.

CSS

Уеб дизайнът на сайта е с отзивчивия веб дизайн (Responsive Web Design- RWD). Дизайнът е оптимизиран според големината на устройството.

```
@media only screen and (max-width: 480px) and (min-width: 320px) {}
```

```
@media only screen and (max-width: 768px) and (min-width: 481px) {}
```

```
@media only screen and (max-width: 1024px) and (min-width: 769px) {}
```

```
@media only screen and (max-width: 1200px) and (min-width: 1025px) {}
```

Javascript

- JavaScript е използван само за активиране на Popup прозорците.
- ```
function togglePopup1(){
document.getElementById("popup-1").classList.toggle("active"); }
```



# Back-end част

# Връзки с front-end частта

- Чрез използването на GET и POST заявки за създава комуникация между потребителя и back-end частта за уебсайта.

```
if (isset($_POST['changeProfile'])) {

 $newUsername = $_POST['changeUsername'];
 $img = "../img" . $_FILES['image']['name'];

 include "../Db/changeUsernameAndProfilePic.php";

 move_uploaded_file($_FILES['image']['tmp_name'], "../img/$img");

}
```

```
if (isset($_GET['accept_id'])) {

 $sender = $user_id;
 $receiver = $_GET['accept_id'];

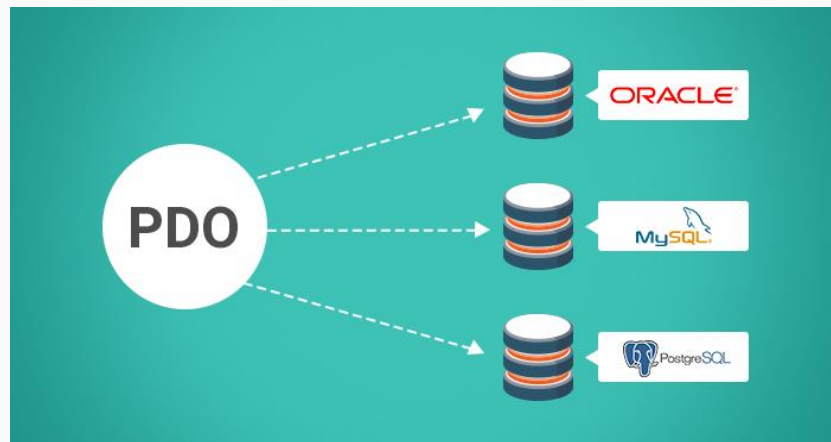
 include "../Db/acceptRequest.php";

}
```

# Връзки с базата данни

За свързването с базата данни сме използва PDO, което предоставя абстракция над базата данни.

В PDO данните са предоставят с параметрични заявки, които предотвратяват SQL инжекции.



# Връзка с база данни

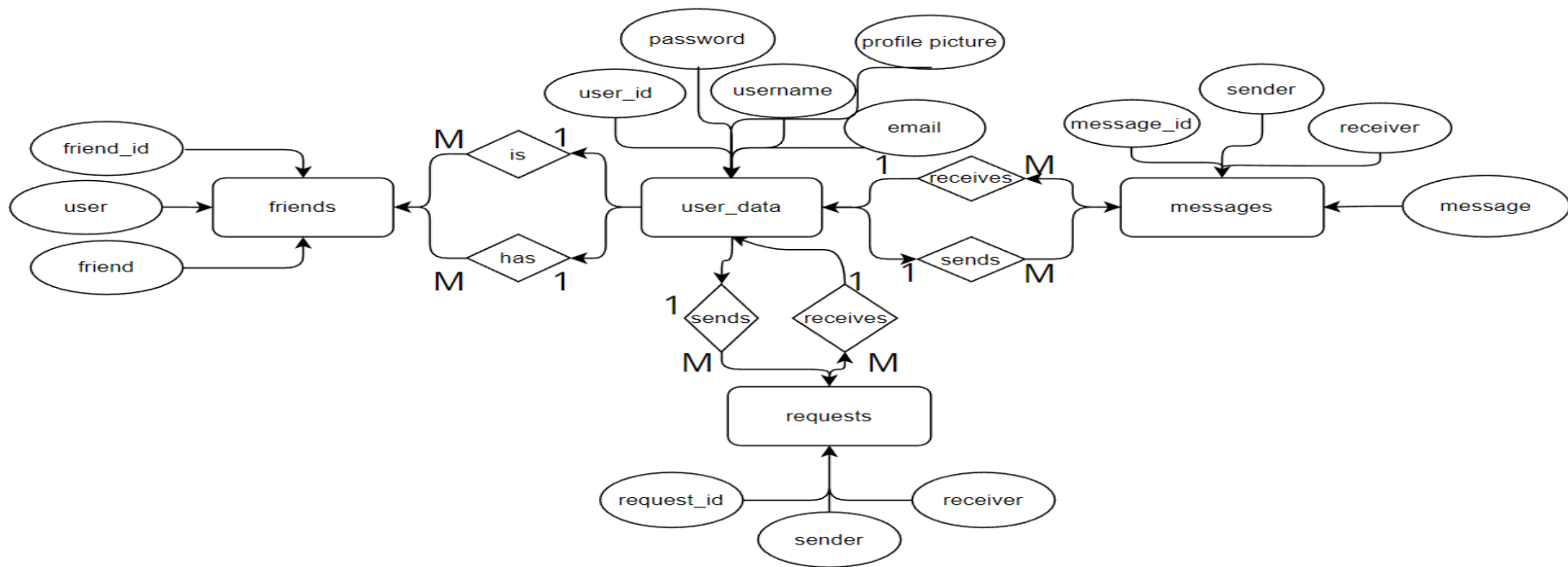
```
$servername = "localhost";
$DBusername = "root";
$DBpassword = "password";
$database = "Curdis";

try {
 $connection = new PDO("mysql:host=$servername;dbname=$database", $DBusername,
$DBpassword);
 $connection->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
} catch(PDOException $e) {
 echo "Connection failed: " . $e->getMessage();
}
```

# Бази данни

- Използвана е релационната база данни MySQL. За СУБД сме използвали приложението с отворен код DBeaver.

# ER диаграмма



# Create заявки

- ```
CREATE TABLE `user_data` (  
  `user_id` int(11) NOT NULL AUTO_INCREMENT,  
  `username` varchar(45) NOT NULL,  
  `email` varchar(45) NOT NULL,  
  `password` varchar(256) NOT NULL,  
  `profilePicture` varchar(45) NOT NULL,  
  PRIMARY KEY (`user_id`),  
  CONSTRAINT `has_friend` FOREIGN KEY (`user_id`) REFERENCES `friends` (`user`),  
  CONSTRAINT `is_friend` FOREIGN KEY (`user_id`) REFERENCES `friends` (`friend`),  
  CONSTRAINT `received_message` FOREIGN KEY (`user_id`) REFERENCES `messages` (`receiver`),  
  CONSTRAINT `sent_message` FOREIGN KEY (`user_id`) REFERENCES `messages` (`sender`),  
  CONSTRAINT `sent_request` FOREIGN KEY (`user_id`) REFERENCES `curdis`.`requests` (`sender`),  
  CONSTRAINT `receoved_request` FOREIGN KEY (`user_id`) REFERENCES `curdis`.`requests` (`receiver`)  
) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci
```


Create заявки

- ```
CREATE TABLE `requests` (
 `request_id` int(11) NOT NULL AUTO_INCREMENT,
 `sender` int(11) NOT NULL,
 `receiver` int(11) NOT NULL,
 PRIMARY KEY (`request_id`),
 KEY `sent_request` (`sender`),
 KEY `received_request` (`receiver`)
) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_general_ci
```

# Create заявки

- ```
CREATE TABLE `messages` (  
  `message_id` int(11) NOT NULL AUTO_INCREMENT,  
  `sender` int(11) NOT NULL,  
  `receiver` int(11) NOT NULL,  
  `message` varchar(1000) NOT NULL,  
  PRIMARY KEY (`message_id`),  
  KEY `sent_message` (`sender`),  
  KEY `received_message` (`receiver`)  
) ENGINE=InnoDB AUTO_INCREMENT=26 DEFAULT CHARSET=utf8mb4  
COLLATE=utf8mb4_general_ci
```

Create заявки

- ```
CREATE TABLE `friends` (
 `friends_id` int(11) NOT NULL AUTO_INCREMENT,
 `user` int(11) NOT NULL,
 `friend` int(11) NOT NULL,
 PRIMARY KEY (`friends_id`),
 KEY `has_friend` (`user`),
 KEY `is_friend` (`friend`)
) ENGINE=InnoDB AUTO_INCREMENT=59 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_general_ci
```

# CRUD заявки

- Дефинира се INSERT заявка на данни в таблицата "messages". Заявката съдържа три параметъра, маркирани със знака "?" - "sender", "receiver" и "message". Тези параметри ще бъдат заменени със стойности при изпълнение на заявката.
- `$sql = "INSERT INTO messages (sender, receiver, message) VALUES (?, ?, ?);";`
- `$connection->prepare($sql)->execute([$messageSender, $messageReceiver, $message]);`

# CRUD заявки

- Използвайки предварително подготвена заявка, кодът извършва изтриване на записи от таблицата "requests" в базата данни. Заявката е дефинирана като "DELETE FROM requests WHERE receiver = ?", като параметърът "receiver" е маркиран със знака "?".
- След това се извиква методът "execute" върху променливата \$sql с масивът [\$user\_id] като аргумент. Това изпълнява заявката за изтриване, като конкретната стойност на "receiver" е заместена със стойността на променливата \$user\_id. Така се осъществява изтриването на записите, които отговарят на зададения "receiver"
- `$sql = $connection->prepare("DELETE FROM requests WHERE receiver = ?");`
- `$sql->execute([$user_id]);`

# CRUD заявки

- Командата използва SQL заявка от типа "SELECT" за избор на стойността на колоната "user\_id" от таблицата "user\_data". Заявката включва условие WHERE, където стойността на параметъра "username" е маркирана със знака "?".
- Методът "execute" се извиква върху променливата \$stmt с масивът [\$username] като аргумент. Това изпълнява заявката, като стойността на "username" е заместена със стойността на променливата \$username.
- След изпълнението на заявката, методът "fetch" се извиква върху променливата \$stmt, за да се вземе следващият ред от резултатите. В този случай, се съхранява стойността на "user\_id" в променливата \$userIdFromName.
- `$stmt = $connection->prepare("SELECT user_id FROM user_data WHERE username = ? ");`
- `$stmt->execute([ $username ]);`
- `$userIdFromName = $stmt->fetch();`

# CRUD заявки

- Кода използва предварително подготвена заявка, за да избере всички данни от таблицата "user\_data", където стойността на колоната "user\_id" съответства на стойността в променливата \$friend['user\_id']. Резултатът се съхранява в променливата \$userById.
- ```
$stmt = $connection->prepare("SELECT * FROM user_data WHERE user_id = ? ");  
$stmt->execute([ $friend['user_id'] ] );  
$userById = $stmt->fetchall();
```


CRUD заявки

- Код използва предварително подготвена заявка за избор на ред от таблицата "user_data", където стойността на колоната "email" съответства на стойността в променливата \$email. Резултатът от заявката се съхранява в променливата \$user.

```
$stmt = $connection->prepare("SELECT * FROM user_data WHERE email = ?");  
$stmt->execute([ $email ]);  
$user = $stmt->fetch();
```

CRUD заявки

Заявката извършва избор на данни от таблиците "messages" и "user_data" чрез предварително подготвена заявка. Заявката избира съобщения и профилни снимки, където "sender" съответства на "user_id" в таблицата "user_data". Резултатът от заявката се съхранява в променливата \$messages.

```
$stmt = $connection->prepare("SELECT messages.message, user_data.profilePicture FROM messages join user_data  
on messages.sender = user_data.user_id where (sender = ? and receiver = ?) or (sender = ? and receiver = ?)");  
$stmt->execute([ $messageSender, $messageReceiver, $messageReceiver, $messageSender ]);  
$messages = $stmt->fetchall();
```

CRUD заявки

- Заявката използва предварително подготвена заявка, за да избере информация за потребители от таблицата "user_data", където "user_id" съответства на стойността в променливата \$user["user_id"]. Заявката извлича "user_id", "username" и "profilePicture". Резултатът от заявката се съхранява в променливата \$friends.

```
$sql = $connection->prepare("select user_id, username, profilePicture from user_data where user_id in  
(SELECT friends.friend FROM Curdis.user_data join friends on user_data.user_id=friends.user where  
user_data.user_id = ?)");  
$sql->execute([ $user["user_id"] ]);  
$friends = $sql->fetchall();
```

CRUD заявки

- Заявката използва предварително подготвена заявка, за да избере информация за потребители от таблицата "user_data", където "user_id" съответства на стойността в променливата \$user["user_id"]. Заявката извлича "user_id", "username" и "profilePicture". При това, се избират само тези потребители, чиито "user_id" се среща в резултатите от подзаявката. Подзаявката избира "user_id" от таблицата "user_data", където "sender" съответства на "user_id" в таблицата "requests", а "receiver" съответства на стойността в променливата \$user["user_id"]. Резултатът от заявката се съхранява в променливата \$requests.

```
$sql = $connection->prepare("select user_id, username, profilePicture from user_data where user_id in (SELECT user_data.user_id FROM Curdis.user_data join requests on user_data.user_id=requests.sender where requests.receiver= ? group by requests.receiver)");  
$sql->execute([ $user["user_id"] ]);  
$requests = $sql->fetchall();
```

CRUD заявки

- Кода използва предварително подготвена заявка, за да избере всички данни от таблицата "user_data", където стойността на колоната "user_id" съответства на стойността в променливата \$messageReceiver. Резултатът от заявката се съхранява в променливата \$currentChat.
- ```
$stmt = $connection->prepare("SELECT * FROM user_data WHERE user_id = ?"); $stmt->execute([
$messageReceiver]); $currentChat = $stmt->fetch();
```

# CRUD заявки

- Кодът използва предварително подготвена заявка, за да избере всички данни от таблицата "user\_data", където стойността на колоната "email" съответства на стойността в променливата \$email и стойността на колоната "password" съответства на стойността в променливата \$hash. Резултатът от заявката се съхранява в променливата \$user.

```
$stmt = $connection->prepare("SELECT * FROM user_data WHERE email = ? AND password = ?");
$stmt->execute([$email, $hash]);
$user = $stmt->fetch();
```

# CRUD заявки

- Код извършва актуализация на данните в таблицата "user\_data". Предварително се дефинира SQL заявка за актуализация, която променя стойностите на колоните "username" и "profilePicture" съответно на стойностите в променливите \$newUsername и \$img, за редовете в таблицата, където стойността на колоната "user\_id" съответства на стойността в променливата \$user\_id, като актуализира редовете в таблицата

```
$sql= "UPDATE user_data set username = ?, profilePicture = ? where user_id = ?";
$connection->prepare($sql)->execute([$newUsername,$img,$user_id]);
```



# CRUD заявки

- Кодът използва предварително подготвена заявка, за да избере всички данни от таблицата "user\_data", където стойността на колоната "email" съответства на стойността в променливата \$email и стойността на колоната "password" съответства на стойността в променливата \$hash. Резултатът от заявката се съхранява в променливата \$user.

```
$stmt = $connection->prepare("SELECT * FROM user_data WHERE email = ? AND password = ?");
$stmt->execute([$email, $hash]);
$user = $stmt->fetch();
```

# CRUD заявки

- Код извършва актуализация на данните в таблицата "user\_data". Предварително се дефинира SQL заявка за актуализация, която променя стойността на колоната "username" на стойността в променливата \$newUsername, за редовете в таблицата, където стойността на колоната "user\_id" съответства на стойността в променливата \$user\_id. Заявката се изпълнява и актуализира съответните редове в таблицата.

```
$sql = "UPDATE user_data set username = ? where user_id = ?";
$connection->prepare($sql)->execute([$newUsername, $user_id]);
```

# CRUD заявки



Кода извършва актуализация на данните в таблицата "user\_data". Предварително се дефинира SQL заявка за актуализация, която променя стойността на колоната "profilePicture" на стойността в променливата \$route, за редовете в таблицата, където стойността на колоната "user\_id" съответства на стойността в променливата \$user\_id. Заявката се изпълнява и актуализира съответните редове в таблицата.

```
$sql = "UPDATE user_data set profilePicture = ? where user_id = ?";
$connection->prepare($sql)->execute([$route, $user_id]);
```

# CRUD заявки

- Код извършва вмъкване на данни в таблицата "requests". Предварително се дефинира SQL заявка за вмъкване, която добавя нов ред в таблицата, като задава стойностите на колоните "sender" и "receiver" съответно на стойностите в променливите \$sender и \$receiver. Заявката се изпълнява и добавя новия ред в таблицата "requests".

```
$sql = "INSERT INTO requests (sender, receiver) VALUES (?,?)";
$connection->prepare($sql)->execute([$sender, $receiver]);
```

# CRUD заявки

- Първата заявка извършва вмъкване на данни в таблицата "friends". SQL заявката задава стойностите на колоните "user" и "friend" съответно на стойностите в променливите \$sender и \$receiver. Заявката се изпълнява и добавя нов ред в таблицата "friends".
- Във втората заявка се извършва още едно вмъкване в таблицата "friends", но с разменени стойности на "user" и "friend". Това е необходимо, за да се добави двустранна връзка между потребителите. Заявката се изпълнява и добавя нов ред в таблицата "friends".
- Третата SQL заявка изтрива редовете, където стойността на колоната "receiver" съответства на стойността в променливата \$sender и стойността на колоната "sender" съответства на стойността в променливата \$receiver. Заявката се изпълнява и изтрива съответните редове от таблицата "requests".

```
$sql = $connection->prepare("insert into friends (user, friend) values (?, ?)");
$sql->execute([$sender, $receiver]);
```

```
$sql = $connection->prepare("insert into friends (user, friend) values (?, ?)");
$sql->execute([$receiver, $sender]);
```

```
$sql = $connection->prepare("DELETE FROM requests WHERE receiver = ? and sender = ?");
$sql->execute([$sender, $receiver]);
```