|  |
| --- |
| DIGITAL  close-up van handen met een schaar en stiften, aan het werkSCRAMBLEPAD |

|  |  |  |
| --- | --- | --- |
| Geïntegreerde proef 2022-2023 |  |  |
| André da Cruz Ribeiro T.I Don Bosco Hoboken |  |  |
|  |  |  |

# voorwoord

Ik presenteer mijn Geïntegreerde Proef over digitale scramblepads als veilig alternatief voor traditionele sloten. In mijn GIP bespreek ik de ontwikkeling van digitale scramblepads.

Ik heb dit project gekozen vanwege de voordelen van digitale scramblepads ten opzichte van traditionele sloten, zoals verbeterde veiligheid, gebruiksgemak en flexibiliteit.

Graag wil ik ook de heer Van Reck en de heer Thijs, die mij gedurende dit project hebben begeleid en waardevolle feedback hebben gegeven.

Ook wil ik graag mijn dank uitspreken aan al mijn klasgenoten die hebben bijgedragen aan de totstandkoming van mijn GIP.

Inhoudsopgave

[voorwoord 3](#_Toc131105948)

[Inleiding 5](#_Toc131105949)

[Software 6](#_Toc131105950)

[Korte uitleg 6](#_Toc131105951)

[Index 8](#_Toc131105952)

[Index 8](#_Toc131105953)

[Login 10](#_Toc131105954)

[Setup 10](#_Toc131105955)

[Scramblepad 11](#_Toc131105956)

[Verificatie 14](#_Toc131105957)

[Deur status 16](#_Toc131105958)

[Administrator 17](#_Toc131105959)

[Index 17](#_Toc131105960)

[Gebruikers toevoegen 17](#_Toc131105961)

[Gebruikers verwijderen 18](#_Toc131105962)

[Deuren toevoegen 19](#_Toc131105963)

[Database aanmaken 20](#_Toc131105964)

[Esp 23](#_Toc131105965)

[Hardware 26](#_Toc131105966)

[BESLUIT 26](#_Toc131105967)

[BIBLIOGRAFIE 26](#_Toc131105968)

[LOGBOEK 26](#_Toc131105969)

# Inleiding

Het doel van mijn GIP is om een veiliger keypad te ontwerpen voor mensen die hun deuren willen beveiligen met een code. Mijn keypad verschilt van de normale keypads doordat de cijfers elke keer van plaats veranderen nadat er een cijfer wordt ingedrukt. Zo kan niemand de code afkijken of raden op basis van de volgorde van de toetsaanslagen

Ik heb de volgende onderdelen gebruiken:

* 2 ESP-32 WROM-DEV
* 2 elektronische sloten
* 2 RGB LEDS

# Software

## Korte uitleg

Voor de ontwikkeling van de digitale scramblepad heb ik een mix van programmeertalen gebruikt, waaronder HTML, CSS, JS, PHP en Ino. Ik koos voor deze talen omdat ik er vertrouwd mee was en omdat ik dacht dat ze goed pasten bij mijn idee voor de scramblepad. Ik heb alle code zelf geschreven, met enige hulp van de heer Van Reck en mijn medestudenten.

Hieronder staat een korte beschrijving van hoe iemand het systeem kan gebruiken.

Voor de initiële setup moet een beheerder op de "admin" knop klikken Chart, treemap chart

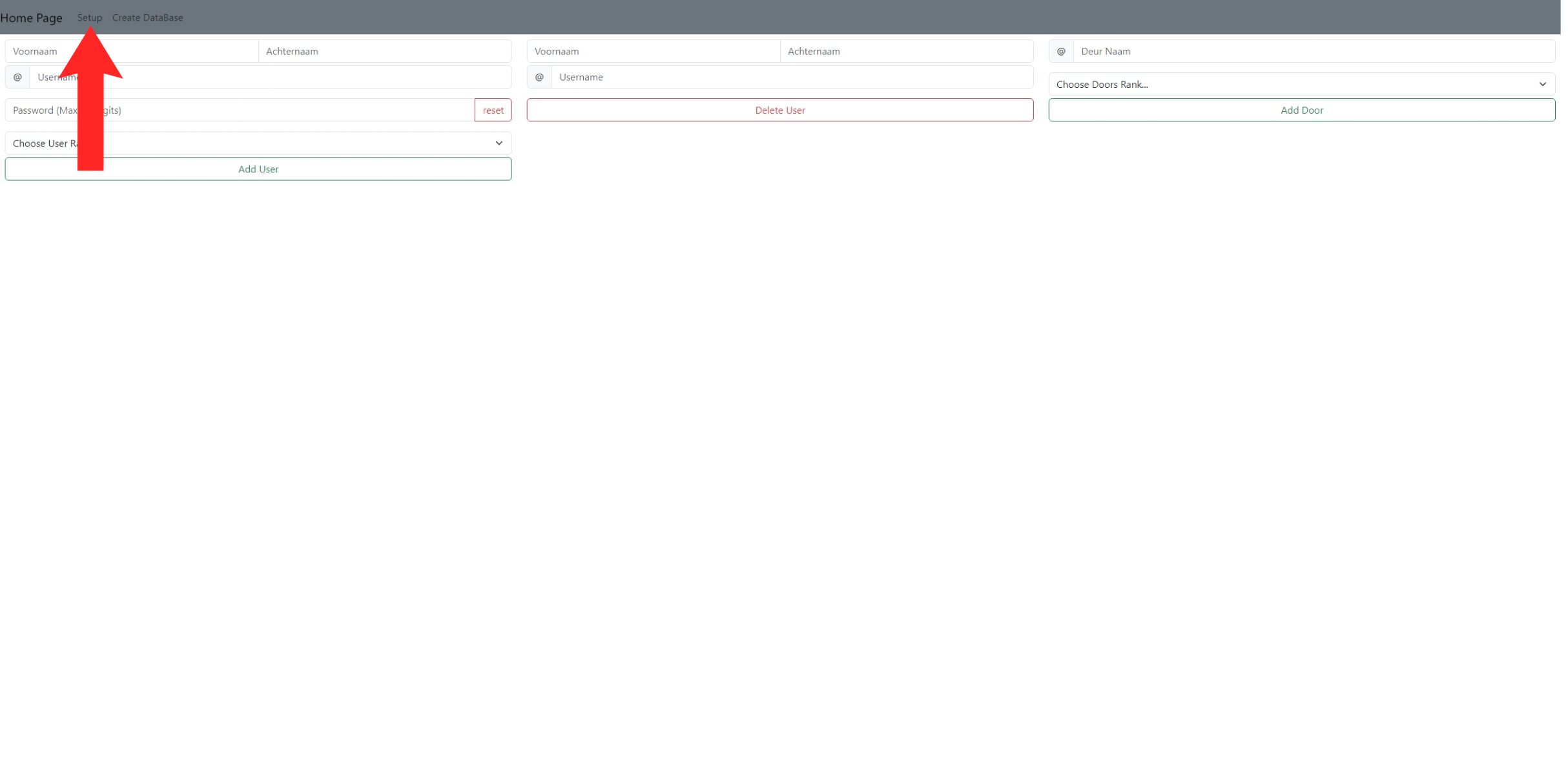
Description automatically generateden vervolgens op "create database" om de benodigde database aan te maken. Graphical user interface, application, Word

Description automatically generated

Daarna kan de beheerder een deur toevoegen aan de database, evenals eventuele andere gebruikers.

Diagram

Description automatically generated with medium confidenceVervolgens kan de beheerder op "setup" klikken, een deur en de rang kiezen.



Nu kunnen de gebruikers zich aanmelden om de deuren te openen.Table

Description automatically generated

Nu zal ik in detail alle scripts en bestanden uitleggen.

## Index

#### Index

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<meta http-equiv="X-UA-Compatible" content="IE=edge" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title>Home Page</title>

<style>

/\* Set body margin and padding to 0 to remove default spacing \*/

body {

margin: 0;

padding: 0;

overflow: hidden;

}

/\* Style the buttons \*/

button {

height: 100%;

border: none;

outline: none;

cursor: pointer;

font-size: 2em;

font-weight: bold;

color: white;

background-color: transparent;

transition: all 0.3s cubic-bezier(0.25, 0.8, 0.25, 1);

}

/\* Style the links inside the buttons \*/

a {

display: block;

text-align: center;

text-decoration: none;

color: inherit;

}

/\* Center the text inside the button and allow clicking anywhere \*/

button a {

display: flex;

justify-content: center;

align-items: center;

height: 100%;

width: 100%;

}

/\* Center the link text inside each button \*/

.left a,

.right a {

display: flex;

justify-content: center;

}

/\* Style the left button \*/

.left {

width: 50%;

float: left;

background-color: green;

}

/\* Style the right button \*/

.right {

width: 50%;

float: right;

background-color: blue;

}

/\* Add a hover effect to the buttons \*/

button:hover {

transform: scale(1.1);

transition: all 0.3s ease-in-out;

}

/\* Adjust hover effect on the left button \*/

.left:hover {

transform: scale(1.1);

transition: all 0.3s ease-in-out;

}

/\* Position the button container \*/

.container {

position: absolute;

top: 0;

bottom: 0;

width: 100%;

}

</style>

</head>

<body>

<!-- Create a container for the buttons -->

<div class="container">

<!-- Create two buttons that split the screen horizontally -->

<button class="left"><a href="admin/index.html">admin page</a></button>

<button class="right"><a href="setup.html">setup page</a></button>

</div>

</body>

</html>

Dit HTML-bestand (index.html) is bedoeld als de startpagina voor een toepassing die een digitale scramblepad gebruikt om deuren te beveiligen. De pagina bevat twee knoppen, een groene en een blauwe, die elk de gebruiker naar een andere pagina leiden.

De groene knop geeft toegang tot de beheerderspagina waar de beheerder nieuwe gebruikers en deuren kan toevoegen of verwijderen en ook de database kan aanmaken.

De blauwe knop leidt naar de instellingspagina waar de beheerder de naam van de deur kan kiezen waarvoor de scramblepad wordt gebruikt.

De code bevat ook CSS-styling om de lay-out van de pagina te bepalen en de knoppen te stijlen. Er wordt gebruik gemaakt van flexbox om de tekst in de knoppen te centreren en overgangseffecten toe te voegen wanneer de gebruiker de muis over de knoppen beweegt. Het doel is om de pagina gemakkelijk te navigeren en de gebruiker de nodige opties te bieden om de digitale scramblepad te configureren en te gebruiken.

## Login

#### Setup

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<meta http-equiv="X-UA-Compatible" content="IE=edge" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title>Setup Page</title>

</head>

<body>

<form action="login/scramblepad.html" onsubmit="setCookies()">

<input

type="text"

name="deurnaam"

id="deurnaam"

placeholder="Deur Naam"

required

/>

<select name="deurrank" id="deurrank">

<option selected>Kies rank</option>

<option value="1">1</option>

<option value="2">2</option>

<option value="3">3</option>

<option value="4">4</option>

</select>

<button type="submit">Finish Setup</button>

</form>

<script>

function setCookies() {

// Get the values of the input and select fields

let deurnaamValue = document.getElementById("deurnaam").value.toLowerCase();

let deurrankValue = document.getElementById("deurrank").value.toLowerCase();

// Set the cookies with the values

document.cookie = "deurnaam=" + encodeURIComponent(deurnaamValue);

document.cookie = "deurrank=" + encodeURIComponent(deurrankValue);

}

</script>

</body>

</html>

Deze code is de HTML en JavaScript code voor de "Setup Page". Het bevat een formulier met drie invoervelden: een tekstveld voor de naam van de deur, een dropdown-menu voor de rank van de deur en een knop om de setup te voltooien. Wanneer de gebruiker op de knop klikt, wordt de JavaScript-functie "setCookies()" opgeroepen, die de waarden van de invoervelden ophaalt en deze waarden gebruikt om cookies te maken voor de naam en de rang van de deur. Deze cookies zullen later gebruikt worden door de "verify.php" pagina, waar de rang van de gebruiker wordt vergeleken met de rang van de deur. Op basis hiervan kan worden bepaald of de gebruiker toestemming heeft om de deur te openen.

#### Scramblepad

###### HTML en JS

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<meta http-equiv="X-UA-Compatible" content="IE=edge" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<link rel="stylesheet" href="scramblepad-style.css" />

<link

href="https://fonts.googleapis.com/css2?family=Bebas+Neue&display=swap"

rel="stylesheet"

/>

<title>test page for scramblepad</title>

</head>

<body>

<div class="parent">

<form action="verify.php" method="post">

<input type="hidden" name="password" id="password" value="" required />

<div class="username-text-field">

<input

class="username-input border"

name="username"

id="username"

placeholder="Username"

aria-label="Username"

aria-describedby="Username"

type="text"

required

/>

</div>

<div class="button-row">

<button class="button border" type="button"></button>

<button class="button border" type="button"></button>

<button class="button border" type="button"></button>

</div>

<div class="button-row">

<button class="button border" type="button"></button>

<button class="button border" type="button"></button>

<button class="button border" type="button"></button>

</div>

<div class="button-row">

<button class="button border" type="button"></button>

<button class="button border" type="button"></button>

<button class="button border" type="button"></button>

</div>

<div class="button-row">

<button class="button border no-invert ❌" type="button">❌</button>

<button class="button border" type="button"></button>

<button class="button border no-invert ✔" type="submit">✔</button>

</div>

</form>

</div>

</body>

</html>

<script>

function shuffleArray(arr) {

arr.sort(() => Math.random() - 0.5);

}

let arr = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9];

shuffleArray(arr);

getKeys();

function getKeys() {

let key = 0;

document.querySelectorAll('button[type="button"]').forEach((element) => {

if (element.textContent == "❌") return;

element.textContent = arr[key];

key++;

});

}

document.querySelector("form").addEventListener("click", (event) => {

if (event.target.textContent == "✔") return;

if (event.target.textContent == "❌") {

document.getElementById("password").value = "";

return;

}

document.getElementById("password").value += event.target.textContent;

shuffleArray(arr);

getKeys();

});

</script>

###### CSS

\* {

box-sizing: border-box;

}

body {

overflow: hidden;

font-family: "Bebas Neue", cursive;

font-size: large;

background-color: rgb(37, 39, 42);

}

.parent {

display: grid;

place-content: center;

height: 100vh;

}

.border {

border-radius: 20px;

border-width: 2px;

border-style: solid;

border-color: black;

font-size: 1.5em;

}

.username-text-field {

padding: 10px;

}

.username-input {

width: 60.5vw;

height: 6vh;

}

.button-row {

padding: 10px;

}

.button {

position: relative;

overflow: hidden;

width: 20vw;

height: 10vh;

text-align: center;

line-height: 50px;

}

.button::before {

content: "";

position: absolute;

top: 0;

left: 0;

width: 100%;

height: 100%;

background-color: transparent;

filter: invert(1);

}

button:not(.no-invert):hover::before {

background-color: rgb(37, 39, 42);

}

.❌ {

background-color: tomato;

}

.❌:hover {

background-color: red;

}

.✔ {

background-color: greenyellow;

}

.✔:hover {

background-color: green;

}

###### Uitleg

Dit is de webpagina waarop de gebruiker zijn code kan invoeren. De knoppen bevatten cijfers van 0 tot 9 die bij elke invoer in een willekeurige volgorde worden geschud.

De code van deze webpagina bestaat uit HTML, CSS en JavaScript. De HTML-code definieert de structuur van de webpagina en de CSS-code geeft de webpagina stijl en opmaak. De JavaScript-code zorgt ervoor dat de knoppen worden geschud.

De JavaScript-code werkt als volgt: wanneer de pagina wordt geladen, wordt er een array gemaakt met de cijfers van 0 tot 9. Vervolgens worden deze cijfers in een willekeurige volgorde geschud met behulp van een functie genaamd shuffleArray(). Deze functie gebruikt de sort() methode van de array met een willekeurige volgorde om de cijfers te schudden.

Vervolgens wordt een functie genaamd getKeys() aangeroepen die de geschudde cijfers toewijst aan de knoppen op de webpagina. Deze functie gebruikt de forEach() methode om door alle knoppen te itereren en de cijfers toe te wijzen op basis van de volgorde van de geschudde array.

Wanneer een gebruiker op een knop klikt, wordt er een event listener geactiveerd die controleert welke knop er is ingedrukt en het bijbehorende cijfer toevoegt aan een inputveld dat is verborgen op de webpagina. De shuffleArray() en getKeys() functies worden vervolgens opnieuw aangeroepen om de knoppen opnieuw te schudden en de nieuwe geschudde cijfers toe te wijzen aan de knoppen.

De CSS-code is verantwoordelijk voor de stijl en opmaak van de webpagina. Het definieert de kleuren, de grootte en de vorm van de knoppen en het inputveld. Het zorgt ook voor animaties en visuele feedback wanneer een gebruiker op een knop klikt.

#### Verificatie

<?php

$link = mysqli\_connect("localhost", "root", "", "locks");

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

$username = $\_POST['username'];

$username = strtolower($username);

$password = $\_POST['password'];

$deurnaam\_cookie = $\_COOKIE['deurnaam'];

$deurnaam\_cookie = strtolower($deurnaam\_cookie);

$deurrank\_cookie = $\_COOKIE['deurrank'];

$sanitized\_username = mysqli\_real\_escape\_string($link, $username);

$sanitized\_password = mysqli\_real\_escape\_string($link, $password);

$sql = "SELECT `rank` FROM `users` WHERE `username`='$sanitized\_username' AND `password`='$sanitized\_password'";

$result = mysqli\_query($link, $sql) or die(mysqli\_error($link));

$user\_rank = mysqli\_fetch\_array($result);

if($user\_rank['rank'] >= $deurrank\_cookie) {

$sql = "UPDATE `doors` SET `status`='open' WHERE `deur naam`='$deurnaam\_cookie'";

$result = mysqli\_query($link, $sql) or die(mysqli\_error($link));

sleep(5); // wait for 5 seconds before updating the status to close

$sql = "UPDATE `doors` SET `status`='close' WHERE `deur naam`='$deurnaam\_cookie'";

$result = mysqli\_query($link, $sql) or die(mysqli\_error($link));

sleep(3);

header("Location: scramblepad.html");

exit();

} else {

// Redirect the user to the scramblepad.html page

header("Location: scramblepad.html");

exit();

}

mysqli\_close($link);

}

?>

Dit stukje PHP code maakt verbinding met een MySQL database genaamd "locks" op de localhost. Vervolgens controleert het of er een POST request is verzonden door de client. Als dat zo is, worden de waarden van de velden "username", "password", "deurnaam" en "deurrank" vanuit de POST data opgeslagen in variabelen. De username en deurnaam worden ook omgezet naar kleine letters om eventuele problemen met hoofdletters te voorkomen.

De functie mysqli\_real\_escape\_string wordt gebruikt om de waarden van de variabelen "username" en "password" te ontsmetten voordat ze worden gebruikt in een MySQL query, om SQL injection aanvallen te voorkomen. Vervolgens wordt er een query uitgevoerd om de rank van de gebruiker op te halen uit de "users" tabel van de database.

Als de rank van de gebruiker gelijk of hoger is dan de rank die is opgeslagen in de "deurrank" cookie, wordt de status van de deur in de "doors" tabel van de database gewijzigd naar "open". Er wordt vervolgens 5 seconden gewacht voordat de status van de deur weer wordt gewijzigd naar "close", dan wordt de gebruiker omgeleid naar de "scramblepad.html" pagina.

Als de rank van de gebruiker lager is dan de rank die is opgeslagen in de "deurrank" cookie, wordt de gebruiker omgeleid naar de "scramblepad.html" pagina.

Als laatste wordt de verbinding met de database gesloten en wordt er geen verdere output gegenereerd.

#### Deur status

<?php

$link = mysqli\_connect("localhost", "root", "", "locks");

$chipid = mysqli\_real\_escape\_string($link, $\_GET["naam"]);

// echo $chipid;

// Controleer of $chipid niet leeg is

if (!empty($chipid)) {

// Controleer of $chipid al in de tabel bestaat

$sql = "SELECT COUNT(\*) FROM doors WHERE `chipid` = '$chipid'";

$result = mysqli\_query($link, $sql) or die(mysqli\_error($link));

$count = mysqli\_fetch\_array($result)[0];

if ($count > 0) { // Als $chipid al in de tabel bestaat

// Haal de status op van de deur met de gegeven $chipid

$sql = "SELECT `status` FROM doors WHERE `chipid` = '$chipid'";

$result = mysqli\_query($link, $sql) or die(mysqli\_error($link));

$status = mysqli\_fetch\_array($result);

// Geef de status terug

echo $status[0];

} else { // Als $chipid niet bestaat in de tabel

// Voeg het nieuwe $chipid toe aan de tabel

$sql = "UPDATE `doors` SET `chipid`='$chipid' WHERE `chipid` = '' LIMIT 1";

mysqli\_query($link, $sql) or die(mysqli\_error($link));

}

} else {

// header('Location: scramblepad.html');

}

?>

Dus, deze pagina is gemaakt om een verzoek van een ESP32 te ontvangen dat zijn ID (chipid) via een GET-request verzendt.

Zodra de pagina het GET-request ontvangt, controleert het of er een geldige chipid in het verzoek aanwezig is. Als er een chipid aanwezig is, controleert de pagina of dit chipid al in de database is opgeslagen. Als dat zo is, haalt de pagina de huidige status van de deur op die aan dit chipid is gekoppeld. Als het chipid nog niet in de database is opgeslagen, voegt de pagina het toe aan de tabel.

Vervolgens stuurt de pagina de status van de deur terug naar de ESP32 als reactie op het GET-verzoek.

## Administrator

#### Index

#### Gebruikers toevoegen

<?php

$link = mysqli\_connect("localhost","root","","locks");

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

$voornaam = $\_POST['voornaam'];

$voornaam =strtolower($voornaam);

$achternaam =$\_POST['achternaam'];

$achternaam =strtolower($achternaam);

$username = $\_POST['username'];

$username = strtolower($username);

$password = $\_POST['password'];

$rank = $\_POST['rank'];

$sql = "INSERT INTO `users`(`voornaam`, `achternaam`, `username`, `password`, `rank`)

VALUES ('$voornaam','$achternaam','$username','$password','$rank')";

$result = mysqli\_query($link, $sql)

or die(mysqli\_error($link));

}

?>

<script>

window.addEventListener("load", () => {

let time = 1

const interval = setInterval(() => {

time--

if(time == 0) {

window.location.replace("index.html")

clearInterval(interval)

}

},1000)

})

</script>

Dit is het stukje PHP-code dat verantwoordelijk is voor het toevoegen van gebruikers.

Het controleert eerst of de HTTP-methode van het verzoek POST is. Vervolgens worden de waarden van de ingediende velden (voornaam, achternaam, gebruikersnaam, wachtwoord en rang) opgehaald en omgezet naar kleine letters met de strtolower()-functie.

Daarna wordt een SQL-query uitgevoerd om de waarden in de database op te slaan met de INSERT INTO-opdracht. Als het invoegen van gegevens in de database mislukt, zal die de foutmelding tonen.

Tot slot, als het invoegen van gegevens succesvol is, wordt de pagina doorverwezen naar "index.html" en wordt er een timer van 1 seconde gestart voordat de doorverwijzing plaatsvindt. Dit wordt bereikt met behulp van de JavaScript-setInterval()-functie.

#### Gebruikers verwijderen

<?php

$link = mysqli\_connect("localhost","root","","locks");

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

$voornaam = $\_POST['voornaam'];

$achternaam = $\_POST['achternaam'];

$username = $\_POST['username'];

echo $voornaam."<br>".$achternaam."<br>".$username."<br>";

$sql = "DELETE FROM `users` WHERE `voornaam`='$voornaam' and `achternaam`= '$achternaam' and `username`='$username'";

$result = mysqli\_query($link, $sql)

or die(mysqli\_error($link));

}

?>

<script>

window.addEventListener("load", () => {

let time = 1

const interval = setInterval(() => {

time--

if(time == 0) {

window.location.replace("index.html")

clearInterval(interval)

}

},1000)

})

</script>

Dit is het stukje PHP-code dat verantwoordelijk is voor het verwijderen van gebruikers. Het script maakt verbinding met de MySQL-database "locks" via de functie mysqli\_connect() en controleert of de aanvraagmethode van het formulier POST is.

Als de aanvraagmethode POST is, worden de waarden van de ingediende formuliervelden 'voornaam', 'achternaam' en 'username' opgehaald en opgeslagen in de corresponderende PHP-variabelen $voornaam, $achternaam en $username.

Vervolgens wordt een DELETE-query uitgevoerd op de tabel 'users' van de database 'locks', waarbij de rij wordt verwijderd waarin de waarden van de kolommen 'voornaam', 'achternaam' en 'username' overeenkomen met de waarden die zijn ingediend via het formulier.

Als de query is voltooid, wordt een JavaScript-script uitgevoerd dat een timer initialiseert die na één seconde de pagina "admin.html" laadt, waardoor de gebruiker terugkeert naar de beheerderspagina.

#### Deuren toevoegen

<?php

$link = mysqli\_connect("localhost","root","","locks");

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

$deurnaam = $\_POST['deurnaam'];

$deurnaam =strtolower($deurnaam);

$rank = $\_POST['rank'];

$sql = "INSERT INTO `doors`(`deur naam`, `status`,`rank`,`chipid`) VALUES ('$deurnaam','close',$rank,'')";

$result = mysqli\_query($link, $sql)

or die(mysqli\_error($link));

}

?>

<script>

window.addEventListener("load", () => {

let time = 1

const interval = setInterval(() => {

time--

if(time == 0) {

window.location.replace("index.html")

clearInterval(interval)

}

},1000)

})

</script>

Dit is het stukje PHP-code dat verantwoordelijk is voor het toevoegen van deuren.

De code maakt verbinding met een MySQL-database en voert vervolgens een INSERT-query uit om de gegevens van de nieuwe deur toe te voegen aan de "doors"-tabel in de database.

De variabelen $deurnaam en $rank worden eerst gevuld met de waarden die in het POST-verzoek zijn ingediend via het index-formulier. De deurnaam wordt vervolgens in kleine letters omgezet met strtolower() en beide waarden worden gebruikt om de SQL-query te genereren die wordt gebruikt om de nieuwe deur toe te voegen aan de database.

Als de query succesvol is uitgevoerd, wordt de gebruiker doorgestuurd naar de "index.html"-pagina met behulp van JavaScript. Als er een fout optreedt bij het uitvoeren van de query, zal de code de foutmelding weergeven die wordt gegenereerd door mysqli\_error($link).

#### Database aanmaken

###### CreateDB.php

<?php

// Database configuration

$host = 'localhost';

$username = 'root';

$password = '';

$database = 'locks';

$sql\_file = 'C:\wamp64\www\GIP\GIP\admin\locks.sql';

// Create connection

$conn = mysqli\_connect($host, $username, $password);

// Check connection

if (!$conn) {

die("Connection failed: " . mysqli\_connect\_error());

}

// Create database if it doesn't exist

$sql = "CREATE DATABASE IF NOT EXISTS $database";

if (mysqli\_query($conn, $sql)) {

echo "Database created successfully!<br>";

} else {

echo "Error creating database: " . mysqli\_error($conn) . "<br>";

}

// Select the database

mysqli\_select\_db($conn, $database);

// Read SQL file

$sql = file\_get\_contents($sql\_file);

// Execute multi-query SQL commands

if (mysqli\_multi\_query($conn, $sql)) {

echo "SQL script executed successfully!";

} else {

echo "Error executing SQL script: " . mysqli\_error($conn);

}

// Close connection

mysqli\_close($conn);

?>

<script>

window.addEventListener("load", () => {

let time = 2

const interval = setInterval(() => {

time--

if(time == 0) {

window.location.replace("index.html")

clearInterval(interval)

}

},1000)

})

</script>

###### Locks.sql

-- phpMyAdmin SQL Dump

-- version 5.2.0

-- https://www.phpmyadmin.net/

--

-- Host: 127.0.0.1:3306

-- Generation Time: Mar 10, 2023 at 03:26 PM

-- Server version: 8.0.31

-- PHP Version: 8.0.26

SET SQL\_MODE = "NO\_AUTO\_VALUE\_ON\_ZERO";

START TRANSACTION;

SET time\_zone = "+00:00";

/\*!40101 SET @OLD\_CHARACTER\_SET\_CLIENT=@@CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET @OLD\_CHARACTER\_SET\_RESULTS=@@CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET @OLD\_COLLATION\_CONNECTION=@@COLLATION\_CONNECTION \*/;

/\*!40101 SET NAMES utf8mb4 \*/;

--

-- Database: `locks`

--

-- --------------------------------------------------------

--

-- Table structure for table `doors`

--

DROP TABLE IF EXISTS `doors`;

CREATE TABLE IF NOT EXISTS `doors` (

`deur naam` varchar(30) NOT NULL,

`status` varchar(7) NOT NULL,

`rank` varchar(10) NOT NULL,

`chipid` varchar(17) NOT NULL

);

-- --------------------------------------------------------

--

-- Table structure for table `users`

--

DROP TABLE IF EXISTS `users`;

CREATE TABLE IF NOT EXISTS `users` (

`voornaam` varchar(30) NOT NULL,

`achternaam` varchar(30) NOT NULL,

`username` varchar(30) NOT NULL,

`password` int NOT NULL,

`rank` varchar(30) NOT NULL

);

--

-- Dumping data for table `users`

--

INSERT INTO `users` (`voornaam`, `achternaam`, `username`, `password`, `rank`) VALUES

('admin', 'admin', 'admin', 0000, '4');

COMMIT;

/\*!40101 SET CHARACTER\_SET\_CLIENT=@OLD\_CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET CHARACTER\_SET\_RESULTS=@OLD\_CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET COLLATION\_CONNECTION=@OLD\_COLLATION\_CONNECTION \*/;

###### Uitleg

Het PHP-script, createDB.php, is verantwoordelijk voor het maken van een MySQL-database en het uitvoeren van een SQL-script (locks.sql) dat de benodigde tabellen en gegevens voor het project bevat. Het begint met het instellen van de databaseconfiguratieparameters zoals de host, gebruikersnaam, wachtwoord en databasenaam.

Vervolgens maakt het een verbinding met de MySQL-server met behulp van de functie mysqli\_connect(). Als de verbinding mislukt, stopt het script en wordt er een foutmelding weergegeven. Als de verbinding succesvol is, gaat het verder met het maken van de database met behulp van de mysqli\_query() functie met de CREATE DATABASE IF NOT EXISTS SQL-opdracht. Als de database met succes is gemaakt, wordt er een bericht weergegeven.

Vervolgens selecteert het script de nieuw gemaakte database met behulp van de functie mysqli\_select\_db(). Vervolgens leest het het SQL-script met behulp van de file\_get\_contents()-functie en voert het uit met behulp van de mysqli\_multi\_query()-functie. Als het SQL-script succesvol is uitgevoerd, wordt er een bericht weergegeven; anders wordt er een foutmelding weergegeven.

Ten slotte verbreekt het script de verbinding met de MySQL-server met behulp van de functie mysqli\_close().

De methode window.addEventListener() in het script is verantwoordelijk voor het omleiden van de gebruiker naar de pagina index.html nadat er 2 seconden zijn verstreken.

Het script locks.sql is een SQL-dumpbestand dat de tabelstructuren en voorbeeldgegevens voor twee tabellen (deuren en gebruikers) bevat. Het begint met het instellen van de SQL-modus en tijdzone. Vervolgens wordt de slotendatabase gemaakt als deze nog niet bestaat.

Het definieert de tabelstructuren voor de deuren en gebruikerstabellen en voegt voorbeeldgegevens in voor de gebruikerstabel. De voorbeeldgegevens bestaan uit één gebruiker met gebruikersnaam admin, wachtwoord 0000 en rang 4.

Ten slotte eindigt het script door de oorspronkelijke tekenset, sortering en tekensetresultaten te herstellen.

## Esp

#include <WiFi.h>

#include <HTTPClient.h>

// #include <Preferences.h>

const char\* ssid = "WIFIIICT";

const char\* password = "fakatijger";

//Your Domain name with URL path or IP address with path

String serverName = "http://10.3.41.25/gip/gip/login/doorStatus.php";

// the following variables are unsigned longs because the time, measured in

// milliseconds, will quickly become a bigger number than can be stored in an int.

unsigned long lastTime = 0;

// Timer set to 10 minutes (600000)

//unsigned long timerDelay = 600000;

// Set timer to 5 seconds (5000)

unsigned long timerDelay = 1000;

const int slot = 12;

const int PIN\_BLUE = 27;

const int PIN\_RED = 26;

const int PIN\_GREEN = 14;

// Preferences preferences;

void setup() {

pinMode(slot, OUTPUT);

pinMode(PIN\_RED, OUTPUT);

pinMode(PIN\_GREEN, OUTPUT);

pinMode(PIN\_BLUE, OUTPUT);

Serial.begin(115200);

WiFi.begin(ssid, password);

Serial.println("Connecting");

while(WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.print(".");

}

Serial.println("");

Serial.print("Connected to WiFi network with IP Address: ");

Serial.println(WiFi.localIP());

Serial.println("Timer set to 5 seconds (timerDelay variable), it will take 5 seconds before publishing the first reading.");

}

void loop() {

setColor(0, 0, 255);

//Send an HTTP POST request every 10 minutes

if ((millis() - lastTime) > timerDelay) {

//Check WiFi connection status

if(WiFi.status()== WL\_CONNECTED){

HTTPClient http;

uint64\_t chipid = ESP.getEfuseMac();

char chipid\_str[17];

sprintf(chipid\_str, "%016llX", chipid);

String serverPath = serverName + "?naam="+chipid\_str;

Serial.println(serverPath);

// Your Domain name with URL path or IP address with path

http.begin(serverPath.c\_str());

// Send HTTP GET request

int httpResponseCode = http.GET();

if (httpResponseCode>0) {

Serial.print("HTTP Response code: ");

Serial.println(httpResponseCode);

String payload = http.getString();

Serial.println(payload);

//check payload (door status)

if (payload == "open" ) {

//open door

setColor(0, 255, 0);

digitalWrite(slot,HIGH);

} else if (payload == "close") {

//close door

setColor(255, 0, 0);

digitalWrite(slot,LOW);

} else if (payload == "wrong"){

setColor(255, 0, 0);

digitalWrite(slot,LOW);

}

} else {

setColor(255, 255, 0);

Serial.print("Error code: ");

Serial.println(httpResponseCode);

}

// Free resources

http.end();

}

else {

setColor(255, 0, 255);

Serial.println("WiFi Disconnected");

}

lastTime = millis();

}

}

void setColor(int R, int G, int B) {

analogWrite(PIN\_RED, R);

analogWrite(PIN\_GREEN, G);

analogWrite(PIN\_BLUE, B);

}

Het programma maakt gebruik van de WiFi- en HTTPClient-bibliotheken om regelmatig een HTTP GET-verzoek te sturen naar een server en te controleren of er specifieke statusinformatie beschikbaar is. Op basis van de ontvangen statusinformatie wordt de kleur van een RGB-LED ingesteld en wordt een digitaal signaal naar een bepaalde uitgangspin van de microcontroller gestuurd om bijvoorbeeld een deur te openen of te sluiten met behulp van een relais.

In de code zijn de variabelen als volgt gedefinieerd:

• 'ssid' en 'password' bevatten de naam en het wachtwoord van het WiFi-netwerk waarop de microcontroller moet inloggen.

• 'serverName' bevat het IP-adres of de domeinnaam van de server en het pad naar de PHP-pagina waarop de statusinformatie beschikbaar is.

• 'lastTime' houdt de tijd in milliseconden bij sinds het laatste HTTP GET-verzoek werd verzonden.

• 'timerDelay' bepaalt de tijd tussen twee opeenvolgende HTTP GET-verzoeken in milliseconden.

• 'slot', 'PIN\_BLUE', 'PIN\_RED' en 'PIN\_GREEN' zijn de uitgangspinnummers van de microcontroller die respectievelijk worden gebruikt om het relais en de RGB-LED aan te sturen.

Een interessant aspect van 'serverName' is dat het de chipid via GET doorstuurt, zodat de server deze kan gebruiken bij de query voor de status. De chipid is een unieke identificatie van de ESP32-module en wordt gegenereerd door de fabrikant. Het bestaat uit een combinatie van een fabrieksgegraveerde identificatie en een uniek serienummer en is 64-bits lang. Het programma gebruikt het EFUSE MAC-adres van de ESP32 als de chipid. Door de chipid als query parameter toe te voegen aan de HTTP GET-aanvraag naar de server, kan de server de ESP32-module identificeren en de status van de deur terugsturen die bij die specifieke module hoort.

# Hardware

Om de deur te openen, gebruik ik hardware zoals een ESP32 microcontroller, een relais om het slot te activeren, en een RGB-LED om de status aan te geven. De gebruiker kan de code invoeren via een tablet die bij de deur is geplaatst of een QR-code scannen met zijn of haar smartphone. Hieronder vindt u foto's van hoe de elektrische slot, ESP32, relais en RGB-LED met elkaar verbonden zijn.

# BESLUIT

# BIBLIOGRAFIE

# LOGBOEK

Ik vind het jammer dat ik niet vanaf het begin van het project GitHub heb gebruikt in plaats van een Excel-sheet. Nadat ik ontdekte hoe gemakkelijk GitHub te begrijpen is, heb ik alles daar geplaatst. Hier is de link naar de GitHub-pagina voor deze GIP: https://github.com/CruZxPog/GIP