



ÓBUDAI EGYETEM

---

Bánki Donát Gépész és Biztonságtechnikai Mérnöki Kar

# Szabaduló szoba vezérlő szoftver tervezése

OE-BGK  
2021.

Hallgató neve:  
Neptun kód:

Kovács Árpád  
BPJZ56

## Tartalomjegyzék

|   |    |
|---|----|
| Jelmagyarázat                                 | 4  |
| Köszönetnyilvánítás                           | 5  |
| 1. Bevezetés                                  | 6  |
| 2. Szakirodalom elemző feldolgozása           | 7  |
| 2.1. Laravel                                  | 7  |
| 2.1.1. A modellek                             | 8  |
| 2.1.2. Nézetek                                | 9  |
| 2.1.3. Kezelő                                 | 9  |
| 2.2. Babel fordító könyvtár                   | 9  |
| 2.3. A Blockly könyvtár                       | 9  |
| 2.4. Raspberry PI 400                         | 9  |
| 2.5. Manjaro ARM                              | 10 |
| 2.6. MySQL                                    | 10 |
| 2.7. PHP                                      | 10 |
| 2.8. NPM/Node.JS                              | 10 |
| 2.9. ESP32                                    | 10 |
| 2.10. RFID olvasó                             | 11 |
| 2.11. OLED kijelző                            | 12 |
| 2.12. Zerotier szolgáltatás                   | 12 |
| 2.13. Fejlesztőkörnyezet - Visual Studio Code | 12 |
| 3. Saját munka                                | 14 |
| 3.1. Adatbázis:                               | 15 |
| 3.2. MVC modell:                              | 17 |
| 3.3. Hálózatkezelés:                          | 21 |
| 3.4. PHP rész:.                               | 21 |

|       |   |    |
|-------|---|----|
| 3.5.  | Vue rész:.                              | 26 |
| 3.6.  | Szabaduló szoba programok programozása: | 27 |
| 3.7.  | Programok futattása:                    | 31 |
| 3.8.  | Kamerák kezelése:                       | 31 |
| 3.9.  | Az eszközök kezelése:                   | 33 |
| 3.10. | Az ESP32 programja:                     | 34 |
| 3.11. | A regisztráció menete:                  | 34 |
| 3.12. | Az eszköz bejelentkezése:               | 35 |
| 3.13. | Relé üzemmód:                           | 36 |
| 3.14. | Bemenet üzemmód:                        | 37 |
| 3.15. | Megjelenítő üzemmód                     | 37 |
| 3.16. | RFID üzemmód                            | 38 |
| 3.17. | Raspberry PI beállítása:.               | 39 |
| 3.18. | A szoftver futattása Raspberry PI-n:.   | 40 |
| 4.    | Összefoglalás                           | 41 |
| 5.    | Idegen nyelvű tartalmi összefoglaló     | 42 |
| 6.    | Irodalomjegyzék                         | 43 |
| 7.    | Mellékletek                             | 44 |
|       | Nyomtatott melléklet                    | 44 |

## Jelmagyarázat

| Jelzés/Rövidítés  | Jelmagyarázat   |
|-------------------|---|
| <b>Laravel</b>    | PHP alapú keretrendszer   |
| <b>Axios</b>      | Egy valós idejű támogatást biztosító könyvtár                   |
| <b>PHP</b>        | Szerveroldali szkriptnyelv dinamikus weblapok készítésére.      |
| <b>JavaScript</b> | Egy böngészőben futó szkript nyelv                              |
| <b>HTML</b>       | HyperText Markup Language                                       |
| <b>CSS</b>        | Cascading Style Sheets  |
| <b>Jetsream</b>   | Laravel Starter kit neve  |
| <b>Tailwind</b>   | CSS könyvtár  |
| <b>Bootstrap</b>  | CSS könyvtár  |
| <b>RFID</b>       | Radio-frequency identification                                  |
| <b>OLED</b>       | Organikus fénykibocsájtó dióda                                  |
| <b>Blockly</b>    | Google által fejlesztett grafikus programozó fejlesztő könyvtár |

## **Köszönetnyilvánítás**

Köszönettel tartozom Dr. Simon János tanár úrnak, a mentorom volt a feladat megvalósítása során. Amikor felötlött bennem a szakdolgozat témája, hozzá fordultam és a véleményét kértem róla. Miután elmagyaráztam, hogy mit is szeretnék akkor alaposan átbeszéltük a részleteket és még bővítettük a már amúgy is vaskos teendők listáját. Megbeszéltük, hogy az elkészült projektnek minden olyan eszközön működnie kell, amelyen a vezető modern böngészők valamely változata fut.

A Rehák kollégium volt és jelenlegi lakóinak.

Valamint köszönettel tartozom még a családomnak, akik nagyban támogattak és hozzájárultak ahhoz, hogy eljuthattam idáig és megírhatam a szakdolgozatomat.

## **1. Bevezetés**

A feladat egy olyan szabaduló szoba vezérlés létrehozása, mellyel akár a programot távolról is lehessen futtatni. A programnak támogatnia kell az ESP32 mikrovezérlőt, valamint tartalmazza, ábrázolja, és engedélyezze a szobák szoftverének szerkesztését.

A feladatot úgy kell megoldani, hogy Raspberry PI mikroszámítógépen is futtatható legyen.

A dokumentum célja a diplomamunka dokumentálása és leírása, a szakdolgozat fő célkitűzése volt, egy olyan szoftver készítése, mely a szabaduló szobáknak a létrehozásában és menedzselésében segít. A szoftvernek támogatnia kell, új szabaduló szobák létrehozását, ESP32 mikrovezérlő szoftverének vezérlését HTTP-protokolon, valamint a szobát fenntartó személyzet segítségét, a csapatok nyomon követését, és nem utolsósorban a szobák programozását.

A feladat eredménye egy olyan alkalmazás létrejötte, mely engedi a felhasználónak, hogy távolról is dolgozhasson.

## 2. Szakirodalom elemző feldolgozása

A szakdolgozat megvalósításához a következő technológiákat használtam fel:

- HTML5
- JavaScript
- Vue.JS
- Axios/Ajax
- Node.js/NPM
- CSS
- Tailwind CSS
- Arduino IDE
- ESP32

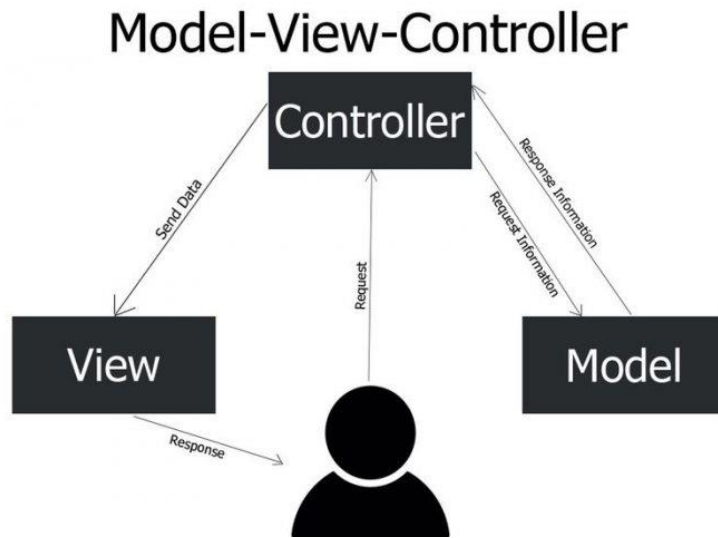
A feladathoz az esp32 típusú mikrovezérlőt lett választva, a feladatot úgy oldottam meg, hogy a program, egy közös csatornán keresztül kommunikál a szerverként üzemelő számítógéppel, és ez úgy lett megvalósítva, hogy azonos hálózathoz, van kötve az összes kliensként dolgozó Raspberry PI 400.

A feladat megoldásához elsődlegesen natív PHP-t terveztem felhasználni, de végül a Laravel nevezetű PHP keretrendszert, mivel biztonságosabb.

### 2.1. Laravel

A Laravel keretrendszer egy Symfony támogatottságra épülő szoftver, mely immáron 8.0-ás verziót is túllépte. A keretrendszer multiplatformos, szóval ugyanúgy fel lehet használni Raspberry PI-ra mint Windows számítógépre.

A keretrendszer csomagoló rendszere Composer névre hallgat, majd felraktam a 8.0-ás Laravelt, és létrehoztam a kezdő fileokat.



*ábra 1 MVC modell*

Ezek végeztével elkezdtem kidolgozni a feladathoz szükséges táblázati elemeket, később feltelepítettem a Laravel Jetstream nevezetű rendszert a keretrendszerbe, e program segítségével elkezdtem a felhasználói felületet, mely Vue.JS Javascript támogatásával üzemel.

Vue 3.0 segítségével elkezdtem az egy oldalas alkalmazás létrehozását, mely csak Ajax használatával kommunikál a rendszerrel, a többit egy előre fordított Javascript-ten keresztül tölti be.

Mivel a Babel könyvtár nem támogatja Bootstrap verziót, ezért szükséges volt a Tailwind CSS könyvtár használatát igénybe vennem.

A modulokat létrehoztam, majd elkezdtem a weboldalhoz tartozó weboldalakat létrehozni. Ezek egy Vue.js könyvtárban találhatóak, melyeknek direkt kommunikációval rendelkeznek a weboldalhoz. Ezt egy úgy nevezett<sup>1</sup> Axios.JS csomagon keresztül intézi.

Mivel a Laravel egy MVC rendszer, ezért szükségesnek érzem el is magyarázni, hogy is dolgozik egy Model-View-Controller struktúra.

### **2.1.1. A modellek**

A modellek amik tárolják adatok struktúráját, amelyek segítségével kommunikálunk az adatbázissal, valamint ezáltal az adatokat írjuk bele.

---

<sup>1</sup> [https://medium.com/@joespinelli\\_6190/mvc-model-view-controller-ef878e2fd6f5](https://medium.com/@joespinelli_6190/mvc-model-view-controller-ef878e2fd6f5)



Erre példa amit használtam programok tárolásánál. Majd ennek a modellhez szükséges kidolgozni az adatbázis felőli kezelő felületét is.

### **2.1.2. Nézetek**

Ebben a projektben ezek az útvonalaknak felelnek meg amit a rendszer felhasznál az előre fordított javascript fileoknál. Normál esetben a lekért útvonalat a szerver maga abban a pillanatban generálja le, és adja át a böngészőnek.

### **2.1.3. Kezelő**

Itt lehetséges a modellek viselkedését meghatározni eseményekre, pl. a lekért modellhez rakjon össze útvonalat és jelenítse meg. Ebben az esetben az Inertia.JS felel a nézet és kontroller közötti kommunikációért.

## **2.2. Babel fordító könyvtár**

Ez a könyvtár lehetővé teszi újabb alkalmazások pl. (Vue, React, vagy Angular) típusú alkalmazások felhasználást régebbi böngészőre, valamint a kódot optimalizálja. Az így létrehozott Stílus és Javascript fájl nem több pár megabyte-nál ezzel levéve a súlyos fileok mozgatását a szerverről. Ezt nevezik még WebPack technológiának is.

## **2.3. A Blockly könyvtár**

Google által fejlesztett grafikus programozási program, mely képes több nyelvre is lefordítani a lehelyezett elemeket, támogatott a php, javascript, go stb.

Az itt felhasznált része annyiban merül ki, hogy a felhasználó ebben rakja össze az ő általa írt programokat, és majd ezeket lementi és lefutattja a kezelő felület.

Ehhez szükséges volt egy külön konzolt létrehozni, mely lehetővé teszi a felhasználóknak a blockly által támogatott blockok futtatását, valamint azok reakcióját a valós rendszerrel.

## **2.4. Raspberry PI 400**

A szolgáltatás futtatására én a Raspberry PI 400-as mikroszámítógépet használtam fel. Ezt használja a felhasználó az eszköz kezelésére. A Raspberry Pi egy bankkártya méretű, egyetlen áramköri lapra/kártyára integrált BCM2835 alapú egykártyás számítógép, amelyet az Egyesült

Királyságban fejlesztettek oktatási célokra. Az eredeti két változat (A és B) kiadása óta már több továbbfejlesztése is kiadásra került. A hivatalosan ajánlott operációs rendszer a laphoz a Raspbian, ami a Debian Linux kifejezetten Raspberry Pi-re optimalizált változata. A legfrissebb modellje a 3. generációs Raspberry Pi 3, de időközben kiadtak még kisebb méretű Zero változatokat is.

## **2.5. Manjaro ARM**

Manjaro ARM olyan operációs rendszer, mely a Raspberry PI 400-on fut. A Manjaro egy Arch típusú Linux disztribúció.

## **2.6. MySQL**

A MySQL egy relációs adatbázis kezelő szoftver.

## **2.7. PHP**

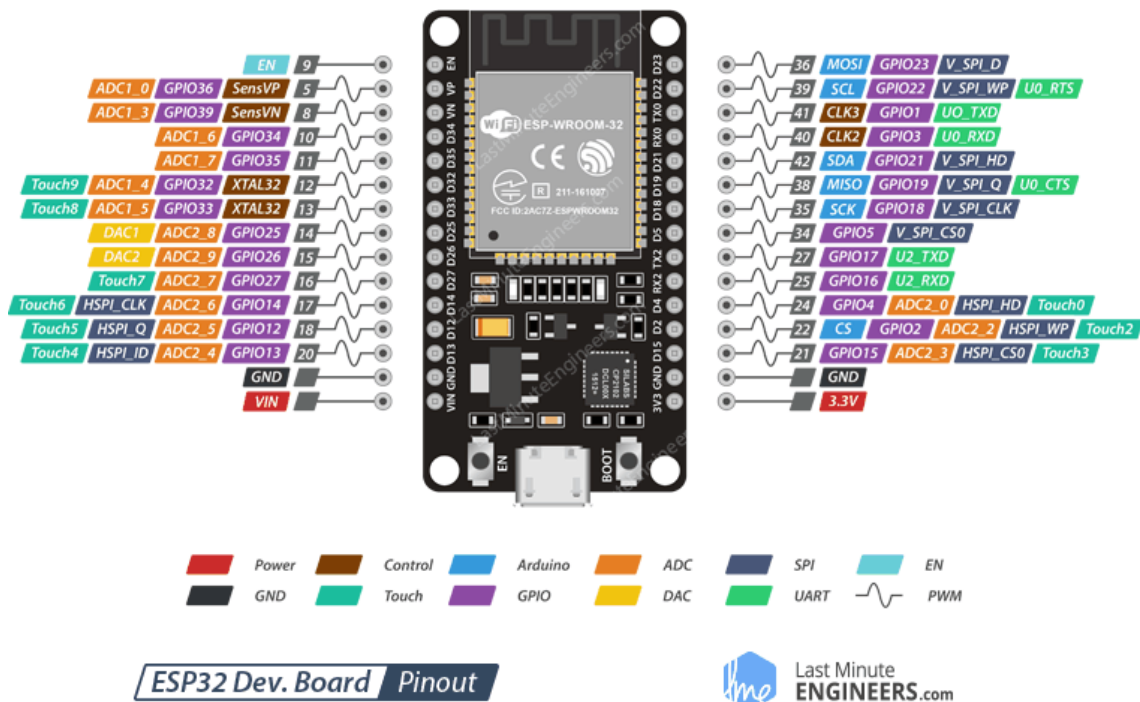
PHP vagyis prehypertext szkript nyelv, ebben íródott a Laravel rendszer.

## **2.8. NPM/Node.JS**

A rendszer az npm könyvtárkezelő segítségével építi fel magát.

## **2.9. ESP32**

Az esp32 az espressif által tervezett és gyártott mikrokontroller. Ezt használtam fel a feladat megoldásához, ez az eszköz kommunikál a Raspberry Pi-vel a Laravel API csatornáján és állítja be a szoftvert.



ábra 2 ESP32

Az ESP32 támogat többféle protokollt, egyelőre 4 dolog lett kidolgozva, képes relé üzemmódban dolgozni, érzékeli azt és elküldi az RFID kártya ID-jét a szervernek, valamint képes bemeneti módban is dolgozni azáltal, hogy a megjelölt (25) portot rövidre zárjuk a földdel. Nem utolsósorban képes az eszköz megjelenítő eszközként is dolgozni amennyiben egy OLED kijelzőt kötünk rá.

## 2.10. RFID olvasó

Az RFID egy rádiós technológián alapuló azonosító eszköz. Különböző eszközök használhatnak RFID tag-eket. S az ESP32 is képes ilyenek olvasására az MFRC522 lappal.

<sup>2</sup> <https://circuits4you.com/2018/12/31/esp32-devkit-esp32-wroom-gpio-pinout/>



ábra 3 MFRC522 RFID olvasó

### 2.11. OLED kijelző

Az OLED kijelző egy grafikus megjelenítő eszköz, melynek segítségével szöveget írhatunk ki az ESP32 eszközről.



5

ábra 4 OLED kijelző

### 2.12. Zerotier szolgáltatás

Egy nyílt forráskódú VPN szolgáltatás, melynek segítségével a fejlesztéshez használt szervergépen lévő MySQL szerver, valamint a Raspberry PI 400 kommunikál. Ehhez létre kellett hozni az ő hálózatukon egy Zerotier network-ot. Majd mind a szervergépet, és a Raspberry PI-t is rácsatlakoztattam.

### 2.13. Fejlesztőkörnyezet - Visual Studio Code

Ez egy a Microsoft által fejlesztett IDE, amely integrált fejlesztő környezetet jelent. A Visual Studio Code talán napjaink egyik legfejlettebb IDE-je, minden eszköz a fejlesztő rendelkezésére áll.

Található benne:

- Debugger

<sup>3</sup><https://github.com/playfultechnology/arduino-rfid-MFRC522>

<sup>5</sup> [https://www.rpibolt.hu/termek/096\\_128x64\\_oled\\_grafikus\\_kijelzo\\_-\\_monokrom\\_feher.html](https://www.rpibolt.hu/termek/096_128x64_oled_grafikus_kijelzo_-_monokrom_feher.html)

- Automatikus kód kiegészítés
- Különféle fejlesztői tesztek
- Szintaxis kiemelés
- Automatikus kódformázás

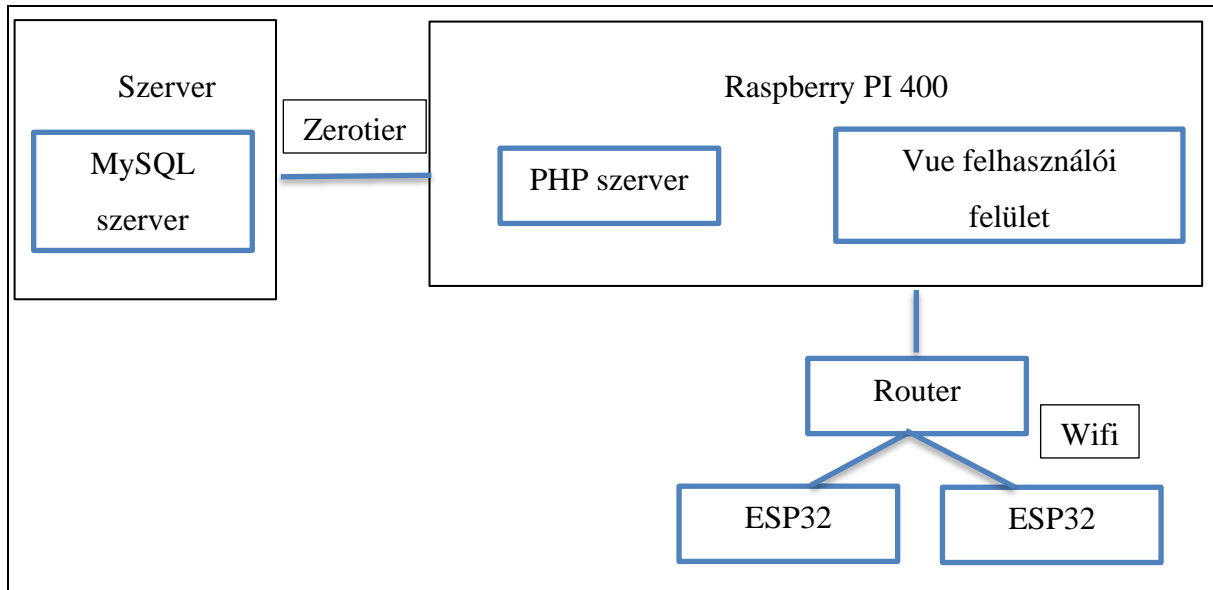
A Visual Studio Code azért is rendkívül jó IDE, mert rengeteg programozási nyelvet támogat. Csak, hogy néhányat említsek:

- JavaScript
- Node.js
- HTML
- CSS
- C#
- C++
- F#
- CoffeeScript
- TypeScript

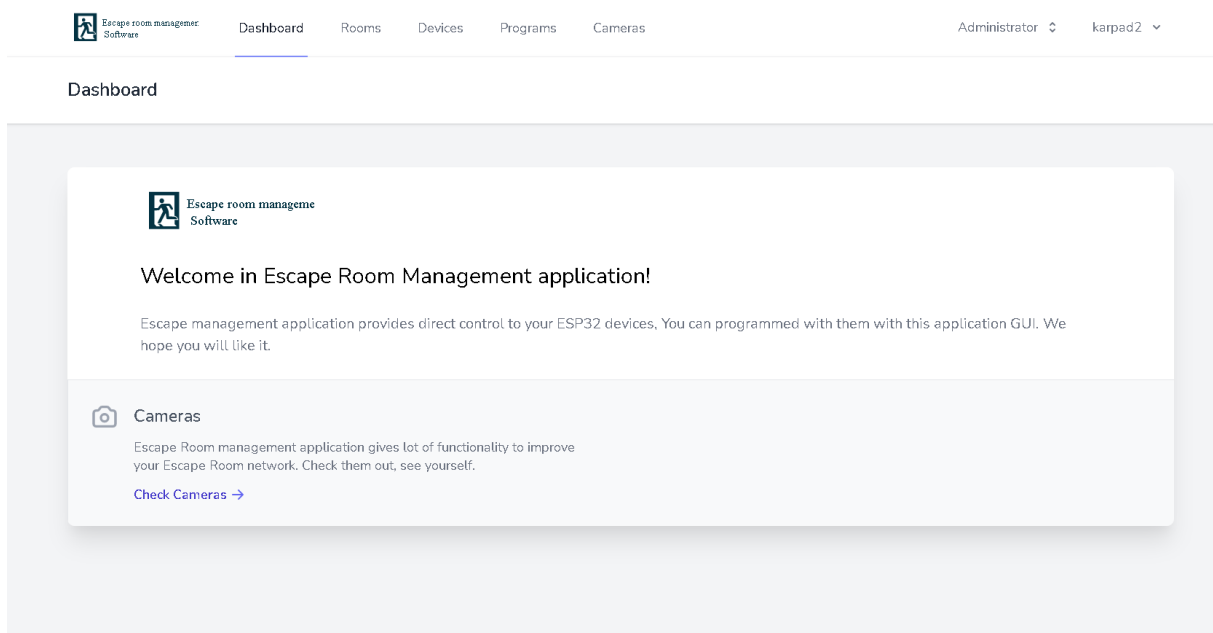
A szakdolgozat megvalósításánál ebben készítettem a JavaScript, Node.js, HTML és CSS részeket.

### 3. Saját munka

Ebben a fejezetben fogom ismertetni a szakdolgozat megvalósítását. A szakdolgozat két részből áll. Az első maga a szerver elkészítése, a második pedig a kliens alkalmazás elkészítése. Mivel mind a két rész igen terjedelmes és összetett, ezért csak a fontosabb részeket fogom ismertetni. A többi rész megtalálható forráskóddal együtt a CD mellékletben. Valamint röviden bemutatom a fejlesztésre használt eszközöket is. Először is talán kezdjük a rendszer blokk vázlatával:



ábra 5 A rendszer blokkvázlata



ábra 6 Az alkalmazás cím oldala

A Laravel keretrendszert választottam majd, elkezdtem létrehozni a kezdő struktúrát, kezdve a felhasználók struktúrájával.

Ehhez a Laravel starterkit-jét telepítettem, mely támogatja a CRFS védelmet, valamint a botok elleni védelmet is, ennek a telepítő kódja:

```
composer require laravel/breeze --dev
php artisan breeze:install
npm install
npm run dev
php artisan migrate
```

Ez a starter kit támogatja az Inertia.JS felhasználását, mellyel egy lapos webalkalmazást lehet létrehozni.

Majd folytatva a beállításokat, beállítottam a levelező szerveret, redis szerveret.

Majd a modellek létrehozását.

```
php artisan make:model Room -a
```

Ez a parancs létrehozza:

- Modellt,
- Migrációs fájlt,
- Gyártó fájlt,
- Vezérlőt.

Mivel a Laravel az egy MVC kategóriába tartozik, ezért szükséges létrehozni a kezelő fájlokat.

### 3.1. Adatbázis:

Rooms table:

```
Schema::create('rooms', function (Blueprint $table) {
    $table->id();
    $table->string('name');
    $table->bigInteger('user_id')->unsigned()->index();
    $table->foreign('user_id')->references('id')->on('users');
    $table->timestamps();
});
```

Devices tábla:

```
Schema::create('devices', function (Blueprint $table) {
    $table->id();
    $table->string('name');
    $table->string('password');
    $table->string('ip_address',45);
    $table->enum('mode',array(["relay","rfid"]));
    $table->integer('status');
    $table->timestamp('last_online');
    $table->bigInteger('room_id')->unsigned()->index()->default('1');
    $table->foreign('room_id')->references('id')->on('rooms');
    $table->timestamps();
});
```

Program tábla:

```
Schema::create('programs', function (Blueprint $table) {
    $table->id();
    $table->string('name');
    $table->boolean('active');
    $table->text('xml_block');
    $table->text('javascript_block');
    $table->bigInteger('room_id')->unsigned()->index()->default('1');
    $table->foreign('room_id')->references('id')->on('rooms');
    $table->timestamps();
});
```

Kamera tábla:

```
Schema::create('cameras', function (Blueprint $table) {
    $table->id();
    $table->string('name')->default('Camera');
    $table->string('url')->default('http://localhost');
    $table->bigInteger('room_id')->unsigned()->index()->default('1');
    $table->foreign('room_id')->references('id')->on('rooms');
```



```
$table->timestamps();

});
```

Teams tábla:

```
Schema::create('teams', function (Blueprint $table) {
    $table->id();
    $table->string('name');
    $table->timestamps();
});
```

Runs tábla:

```
Schema::create('runs', function (Blueprint $table) {
    $table->id();
    $table->bigInteger('room_id')->unsigned()->index();
    $table->foreign('room_id')->references('id')->on('rooms');
    $table->bigInteger('program_id')->unsigned()->index();
    $table->foreign('program_id')->references('id')->on('programs');
    $table->bigInteger('team_id')->unsigned()->index();
    $table->foreign('team_id')->references('id')->on('teams');
    $table->timestamp('start_time')->default(now());
    $table->timestamp('finish_time')->nullable();
    $table->timestamps();
});
```

Ezáltal létrehoztuk a struktúrát.

A következő paranccsal meg lementjük a struktúrát az adatbázisba.

```
php artisan migrate
```

A php artisan leképezi a generált fájlokat a .env fileban található adatbázisba.

Majd a kezdő (alap) sorokat létrehozza, melyekre később lesz hivatkozva.

### 3.2. MVC modell:

Ez alapján kezdtem el kidolgozni az MVC modellt. Meg kell említeni, hogy a rendszer elsődlegesen AJAX adatkommunikációt használ, míg a programkínézetek előre vannak fordítva az app.js file-ba, a Babel könyvtár segítségével.

Modellel kezdtem.

Szoba tábla modellje:

```
class Room extends Model
{
  use HasFactory;
  protected $fillable=['name','user_id'];
}
```

Teams tábla modellje:

```
class Teams extends Model
{
  use HasFactory;
  protected $fillable=['name'];
}
```

Runs tábla modellje:

```
class Runs extends Model
{
  use HasFactory;
  protected $fillable=['team_id','program_id','start_time'];
}
```

Device tábla modellje:

```
class Device extends Model
{
  use HasFactory;
  protected $fillable = [
    'name',
    'mode',
    'status',
    'text',
    'room_id'
  ];
  protected $hidden=[
    'ip_address',
    'password'
  ];
}
```

User tábla modellje:

```
class User extends Authenticatable
{
  use HasFactory, Notifiable;

  protected $fillable = [
    'name',
    'email',
    'password',
  ];

  protected $hidden = [
    'password',
    'remember_token',
  ];

  protected $casts = [
    'email_verified_at' => 'datetime',
  ];
}
```

Programs tábla modellje:

```
class Programs extends Model
{
  use HasFactory;
  protected $fillable = [
    'name',
    'active',
    'javascript_block',
    'room_id'
  ];
  protected $hidden=[
    'xml_block',
  ];
}
```

Cameras tábla modellje:

```
class Camera extends Model
{
  use HasFactory;
  protected $fillable = [
    'name',
    'url',
    'room_id'
  ];
}
```

Ezek végeztével elkezdtem kidolgozni a nézeteket is. Ezeknek a használatához igénybe vettem a Vue.JS könyvtárat. Majd párhuzamosan kidolgoztam az útvonalat (Route-t) és a hozzá tartozó nézetet. Mivel a cél az egy oldalas alkalmazás készítése volt, ezért segítségül vettem megint az Inertia.JS könyvtárat és az ahhoz tartozó tábla kezelő modulokat is.

Ezt a modult feltelepítve létrehoztam a következő rendszert.

Az Api.php route-k, melyben főleg az esp-k útvonalai találhatók.

```
<?php
use Illuminate\Http\Request;
use App\Models\Devices;
use App\Models\Room;
use App\Models\Cameras;
use Illuminate\Support\Facades\Route;
use Illuminate\Support\Facades\Auth;

Route::get('/', function (Request $request) {
    return Room::join('cameras', 'rooms.id', '=', 'cameras.room_id')->where('user_id', 1)->get();
});
Route::get('device/add-device', function (Request $request) {
    $id=Devices::create(['name'=>'Default', 'ip_address'=>$request->getClientIp(), 'mode'=>'relay', 'status'=>'0', 'last_online'=>now(), 'room_id'=>1, 'password'=>'asdf'>id;
    Devices::findOrFail($id)->update(['password'=>md5('salt'.$id.'salt')]);
    return Devices::findOrFail($id);
});
Route::get('device/status/{device_password}', function ($device_password) {
    $device=Devices::where('password', $device_password)->get();
    Devices::findOrFail($device[0]->id)->update(['last_online'=>now()]);
    return $device[0];
});
Route::get('device/input/{device_password}/{status}', function ($device_password, $status) {
    $device=Devices::where('password', $device_password)->update(['status'=>$status]);
    Devices::findOrFail($device[0]->id)->update(['last_online'=>now()]);
    return $device[0];
});
Route::get('device/rfid/{device_password}/{code}', function ($device_password, $code) {
    $device=Devices::where('password', $device_password)->update(['text'=>$code]);
    Devices::findOrFail($device[0]->id)->update(['last_online'=>now()]);
    return $device[0];
});
Route::get('device/store/dev-api/{device_password}/{status}', function ($device_password, $status) {
    $device= Devices::where('password', $device_password)->get();
    Devices::findOrFail($device[0]->id)->update(['status'=>'1']);
    return 'ok';
});
Route::get('device/js-api/{id}/{mode}/{status}', function ($id, $mode, $status) {
    $device= Devices::findOrFail($id)->get();
    Devices::findOrFail($device[0]->id)->update(['mode'=>$mode, 'status'=>$status]);
    return 'ok';
});
Route::get('device/all', function ($id, $mode, $status) {
```

```

        return Devices::all();
    });

Route::middleware('auth:sanctum')->get('/user', function (Request $request) {
    return $request->user();
});

Route::middleware(['auth:sanctum', 'verified'])->post('update-program/{program_id}', function (Re
    $request, $program_id) {

        Programs::where('id', $program_id)->update(['name'=>$request-
>name, 'active'=>1, 'javascript_block'=>$request->javascript_block, 'xml_block'=>$request->xml_bloc
    });

```

### 3.3. Hálózatkezelés:

Az alapötlete a szervernek és a használatának a peertől-peerig való használata. Ez magába foglalja az adatok védelmét is, mivel az adatok 80-as porton utaznak, ezért szükséges volt védelmet használni, ezért szükséges volt bevezetni a Raspberry Pi-t, mint az esp32 hálózatnak a „router-ét” amivel eléri a szerver, ehhez én a Zerotier nevezetű nyílt forrású VPN szolgáltatás vettem fel, mellyel a felhasználó hozzáférést az eszközhöz LAN kapcsolaton keresztül a hálózathoz, így könnyedén a többi eszköz felcsatlakozhat a zerotier vpn-jéhez, amin keresztül az esp32 mikrovezérlők hozzáférnek a szerverhez. Valamint hálózati híd létrehozva lehet kapcsolódni a vezérlő weboldalhoz is melynek a címe <http://192.168.193.234>.

A szerveren található egy MySQL adatbázis amihez kapcsolódnak a Raspberry Pi-k. Ezáltal segítve a fejlesztésben, hogy ne külön bázisokkal dolgozzak.

### 3.4. PHP rész:.

A Laravel alapértelmezetten Vue.JS keretrendszert használja. Ezért az útvonalakat is php-ban, valamint kinézeti file-okat Vue-ban kellett megírni.

A web.php tartalma, mely tartalmazza az útvonalakat, és a szükséges renderelő fájlokat.

```

<?php

use Illuminate\Foundation\Application;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Route;
use Illuminate\Support\Facades\Auth;
use Inertia\Inertia;
use App\Models\Devices;
use App\Models\Room;

use App\Http\Controllers\DevicesController;
use App\Http\Controllers\CamerasController;

```

```

use App\Http\Controllers\EteamsController;
use App\Http\Controllers\ProgramsController;
use App\Http\Controllers\RoomController;
use App\Models\Cameras;
use App\Models\ETeams;
use App\Models\Programs;
use App\Models\Run;

//use GuzzleHttp\Psr7\Request;

/*
|-----
| Web Routes
|-----
|
| Here is where you can register web routes for your application. These
| routes are loaded by the RouteServiceProvider within a group which
| contains the "web" middleware group. Now create something great!
|
*/

$id=Auth::id();
/*Tests*/
Route::get('test',function()
{
    return 'k';
});
/*End Tests*/

/*Inertia pages*/
Route::get('/',
function () {
    return Inertia::render('Welcome', [
        'canLogin' => Route::has('login'),
        'canRegister' => Route::has('register'),
        'laravelVersion' => Application::VERSION
    ]);
});

Route::middleware(['auth:sanctum', 'verified'])->
get('/dashboard', function () {
    return Inertia::render('Dashboard');
})->name('dashboard');

Route::middleware(['auth:sanctum', 'verified'])->
get('/rooms', function () {
    return Inertia::render('rooms',[
        'rooms'=> Room::where('user_id',Auth::id())->get()
    ]);
})->name('rooms');

Route::middleware(['auth:sanctum', 'verified'])->
get('/room/{room_id}', function ($room_id) {
    return Inertia::render('room',[
        'room'=>Room::where('id',$room_id)->get()
    ]);
})->name('room');

Route::middleware(['auth:sanctum', 'verified'])->
get('/devices', function () {
    return Inertia::render('devices',[
        'devices'=>Room::join('devices','rooms.id','=','devices.room_id')->
where('user_id',Auth::id())->get()

```

```

    });
  })->name('devices');

Route::middleware(['auth:sanctum', 'verified'])->
get('/device/{device_id}', function ($device_id) {
    return Inertia::render('device', [
        'device'=> Devices::where('id',$device_id)->get()
    ]);
  })->name('device');

Route::middleware(['auth:sanctum', 'verified'])->
get('/programs', function () {
    return Inertia::render('programs', [
        // 'rooms'=> Room::where('user_id',Auth::id())->get(),
        'programs'=>Room::join('programs','rooms.id','=', 'programs.room_id')->
>where('user_id',Auth::id())->get()
        //join('rooms','programs.room_id','=', 'room.id')->
>where(['user_id',Auth::id()])->
    ]);
  })->name('programs');

Route::middleware(['auth:sanctum', 'verified'])->
get('/program/{program_id}', function ($program_id) {
    return Inertia::render('program', [
        'program'=>Programs::findorFail($program_id)
    ]);
  })->name('program');

Route::middleware(['auth:sanctum', 'verified'])->
get('/cameras', function () {
    return Inertia::render('cameras', [
        'cameras'=>Room::join('cameras','rooms.id','=', 'cameras.room_id')->
>where('user_id',Auth::id())->get()//->where(['user_id', Auth::id()])
    ]);
  })->name('cameras');

Route::middleware(['auth:sanctum', 'verified'])->
get('/rteams', function () {
    return Inertia::render('teams', [
        'teams'=>ETeams::all()
    ]);
  })->name('teams');
/*End Inertia pages*/

/* Devices*/
Route::middleware(['auth:sanctum', 'verified'])->
post('add-user-device',
function (Request $request){
    $id=Auth::id();
    Devices::where('password',$request->device_id)->where('user_id',1)-
>update(['user_id',$id]);
    return 'ok';
});

Route::middleware(['auth:sanctum', 'verified'])->
get('assign-device/{dev_id}/{room_id}',function ($dev_id,$room_id){
    Devices::findorFail($dev_id)->update(['room_id',$room_id]);
    return 'ok';
});

Route::middleware(['auth:sanctum', 'verified'])->
get('assign-device/{dev_id}/{room_id}',function ($dev_id,$room_id){
    Devices::findorFail($dev_id)->update(['room_id',$room_id]);

```

```

    return 'ok';
  });
  /*End Devices*/
  /*Rooms*/
  Route::middleware(['auth:sanctum', 'verified'])->
  post('create-room',
  function (Request $request){
    Room::create(["name"=>$request->name,"user_id"=>Auth::id()]);
    return 'ok';
  });

  Route::middleware(['auth:sanctum', 'verified'])->
  post('update-room',function (Request $request,$room_id){
    Room::create(["name"=>$request->name,"user_id"=>Auth::id()]);
    return 'ok';
  });
  /*End programs*/

  /*Programs*/
  Route::middleware(['auth:sanctum', 'verified'])->
  get('create-program/{room_id}',function (Request $request,$room_id){
    return Inertia::render('program',["program"=>null]);
  })->name('program');

  Route::middleware(['auth:sanctum', 'verified'])->
  post('create-program/{room_id}',function (Request $request,$room_id){
    //Programs::where(['room_id',$room_id])->update(['active'=>0]);
    Programs::create(['name'=>$request->name,'active'=>1,'javascript_block'=>$request->javascript_block,'xml_block'=>$request->xml_block,'room_id'=>$room_id]);
    return 'ok';
  });

  Route::middleware(['auth:sanctum', 'verified'])->
  get('update-program/{program_id}',function (Request $request,$program_id){
    return Inertia::render('program',[
      'program'=> Programs::where('id',$program_id)->get()
    ]);
  })->name('program');

  Route::middleware(['auth:sanctum', 'verified'])->
  post('update-program/{program_id}',function (Request $request,$program_id){

    Programs::where('id',$program_id)->
    update(['name'=>$request->name,'active'=>1,'javascript_block'=>
    $request->javascript_block,'xml_block'=>$request->xml_block]);

  });
  /*End Programs*/

  /*Cams*/
  Route::middleware(['auth:sanctum', 'verified'])->
  post('add-camera-room/{room_id}',function (Request $request,$room_id){
    Camera::create(['name'=>$request->camera_name,'url'=>$request->camera_url,'room_id'=>$room_id]);
    return 'ok';
  });

  Route::middleware(['auth:sanctum', 'verified'])->
  post('update-cam/{camera_id}',function (Request $request,$camera_id){
    Camera::findorFail($camera_id)->update(['name'=>$request->name,'url'=>$request->camera_url]);
    return 'ok';
  });

```



```

/*End Cams*/

/*Teams*/
Route::middleware(['auth:sanctum', 'verified'])->
get('create-rteam',function (Request $request){
    ETeams::create(['name'=>$request->name]);
    return 'ok';
});
Route::middleware(['auth:sanctum', 'verified'])->
post('update-rteams/{teams_id}',function (Request $request,$teams_id){
    Cameras::findorFail($teams_id)->update(['name'=>$request->name]);
    return 'ok';
});

/*End Teams*/

Route::middleware(['auth:sanctum', 'verified'])->
get('run/{room_id}',function ($room_id){
    $room=Room::findorFail($room_id)->get();
    $program=Programs::where(['room_id',$room_id],['active',1])->get();

    $active_run = Run::where(["room_id",$room_id],["finish_time",NULL])->get();
    if($active_run->isEmpty()){
        {
            $active_run=Run::create(["room_id"=>$room_id,"program_id"=>$program-
>id,"team_id"=>1,"start_time"=>now(),"finish_time"=>null])->get();
        }

        return Inertia::render('run',[
            'room'=>$room,
            'program'=>$program,
            'run'=>$active_run
        ]);
    }->name('run');

Route::middleware(['auth:sanctum', 'verified'])->
get('run/{room_id}/stop',function ($room_id){
    $room=Room::findorFail($room_id)->get();
    $program=Programs::where(['room_id',$room_id],['active',1])->get();

    $active_run = Run::where(["room_id",$room_id],["finish_time",NULL])->
update(["finish_time",now()]);
    return Inertia::render('run',[
        'room'=>$room,
        'program'=>$program,
        'run'=>$active_run
    ]);
}->name('run');

```

### 3.5. Vue rész:.

A Vue kinézeti fileok a következő képen épülnek fel:

```
<template>
Itt található a Vue elemekből összeállított modell (Html)
</template>
<script>
Itt funckionalitást rakjuk össze.(JS)
export default {
  components: {
  },
  data: ()=>
  {
    Itt találhatóak program változói,
  },
  methods:
  {
    program()
  },
  mounted: mounted()
  {
    Meghíváskor lefutó szkript
  }
}
</script>
<style>
Míg a stílusban kinézeti css-t állítjuk be. (CSS)
</style>
```

A felhasznált szkriptek a következő képen vannak eltárolva:

- **Komponensek:** több azonos modult felhasználó elemeket tárolok, itt vannak Jetstream eredeti filejai is pl. Checkbox.vue.
- **Az oldalak:** Létre van hozva egy Pages nevű mappa ott találhatóak Vue oldalak, ami komponensekből áll.
- **Rendszert konfiguráló beállítások:** Ezek azok a fileok amik meghatározzák az alábbi fileok működését, pl. Itt van meghatározva, hogy az elemek Tailwind css-t használnak.

A program kiinduló pontja:

```
- const el = document.getElementById('app');

createApp({
  render: () =>
    h(InertiaApp, {
      initialPage: JSON.parse(el.dataset.page),
      resolveComponent: (name) =>
require(`./Pages/${name}`).default,
    }),
  .mixin({ methods: { route } })
  .use(InertiaPlugin)
  // .use(BootstrapVue)
  // .use(IconsPlugin)
  // .config.devtools = true
  .mount(el);

InertiaProgress.init({ color: '#043343' });
```

### 3.6. Szabaduló szoba programok programozása:

A programnak szükséges támogatnia a Blockly könyvtárat, melynek fő használata program.vue fileon belül található.

```
Route::middleware(['auth:sanctum', 'verified'])->get('update-
program/{program_id}',function (Request $request,$program_id) {
  return Inertia::render('program',[
    'program'=> Programs::where('id',$program_id)->get()
  ]);
})->name('program');
```

Amit a rendszeren keresztül meg kap a vue.js.

```
<template>
  <app-layout>
    <template #header>
      <h2 class="font-semibold text-xl text-gray-800 leading-
tight">
        Programming Interface {{program[0].name}}
      </h2>
    </template>
    <div class="py-12">
      <div class="max-w-7xl mx-auto sm:px-6 lg:px-8">
        <div class="bg-white overflow-hidden shadow-xl sm:rounded-
lg">

          <div class="col-span-6 sm:col-span-4">
            <jet-label for="program_name" value="Program Name" />
            <jet-input id="program_name" type="text" class="mt-1 block w-
```

```

full" v-model="program[0].name" ref="program_name"
autocomplete="program_name" />
    <jet-input-error :message="program_name" class="mt-2" />
</div>
    <div id="blocklyDiv" class="mt-1 block w-full" style="height:
480px;"></div>
    <block_vue />
    <xml id="blocklyDefault" v-model="program[0].xml_block"
style="display: none"></xml>
    <textarea id="program_javascript" v-
model="program[0].javascript_block" class="mt-1 block w-full"
style="display: block" name="program_javascript"></textarea>
    <textarea id="program_xml" v-model="program[0].xml_block"
name="program_xml" style="display: none"></textarea>
    </div>
    <jet-button v-on:click="save">
        Save
    </jet-button>
</div>
</app-layout>
</template>

<script>
import AppLayout from '@/Layouts/AppLayout'
import Welcome from '@/Jetstream/Welcome'
import block_vue from '@/Components/block_vue';
import JetActionMessage from '@/Jetstream/ActionMessage'
import JetButton from '@/Jetstream/Button'
import JetFormSection from '@/Jetstream/FormSection'
import JetInput from '@/Jetstream/Input'
import JetInputError from '@/Jetstream/InputError'
import JetLabel from '@/Jetstream/Label'
import axios from 'axios';
import Blockly, { Workspace } from 'blockly';
import * as En from 'blockly/msg/en';
import BlocklyJS from 'blockly/javascript';

export default {
    components: {
        AppLayout,
        block_vue,
        Welcome,
        JetButton,
        JetInput,
        JetInputError,
        JetLabel,
        Blockly
    },
    data() {
        return {
            a_program_name: '',
            a_program_xml: '',
            a_program_javascript: '';
        },
        props: {
            program: {

```

```

        type:Array,
        required: true
    },
    },
    methods:{
        myUpdateFunction(event) {},
        save() {
            axios.post("/api/update-
program/"+this.program[0].id,{
                name:this.program[0].name,
                xml_block:this.program[0].xml_block,
                javascript_block:this.program[0].javascript_block});
        },
        auto_compile(workspace)
        {
            console.log("im here")
        }
        this.program[0].javascript_block=BlocklyJS.workspaceToCode(workspace);
        let xml = Blockly.Xml.workspaceToDom(workspace);
        this.program[0].xml_block=xml;
    }
    },
    mounted () {
        let defaultBlocks = document.getElementById('blocklyDefault');
        Blockly.setLocale(En);
        let Workspace = Blockly.inject('blocklyDiv',{
            media: '/media/',
            toolbox: document.getElementById('toolbox')});
        Workspace.addChangeListener(()=>
        {
            this.auto_compile(Workspace)
        });

        Blockly.Xml.domToWorkspace(defaultBlocks, Workspace);
    }
}
</script>
<style>

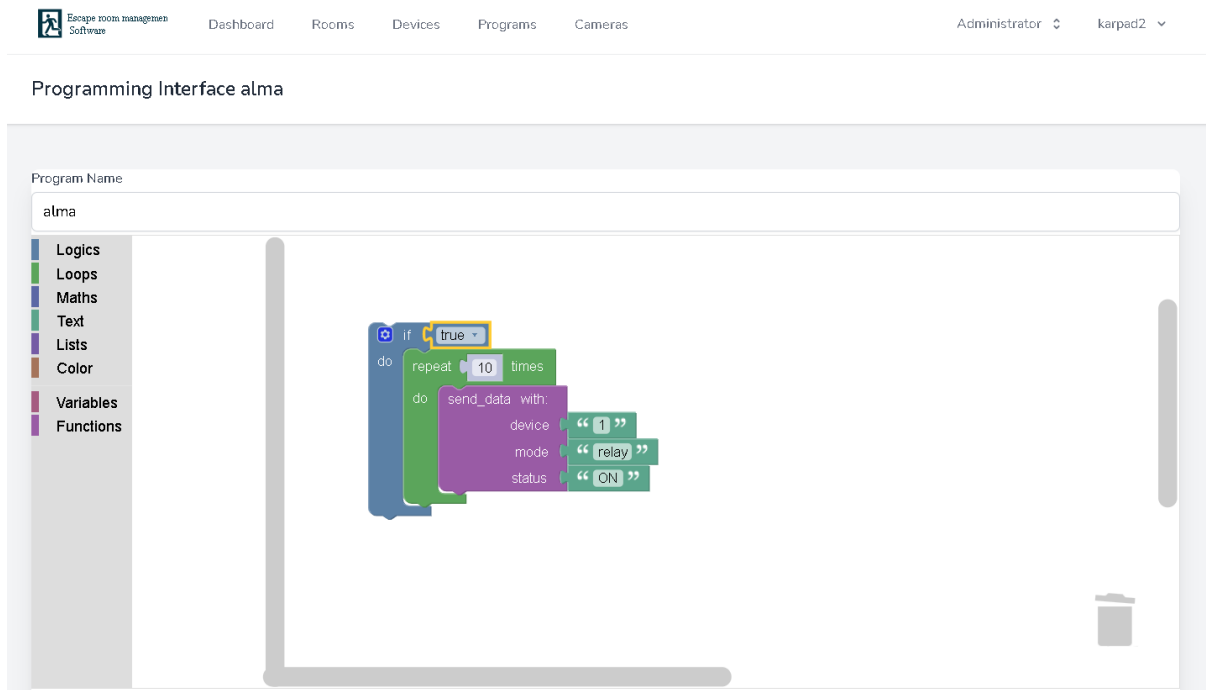
</style>

```

A program átadja az xml, és a javascript változatát a programnak, ezt a Blockly felhasználja és a Vue beilleszti a két kijelölt textarea html tag-be. Minden egyes változtatásnál a felületen a Blockly.JS nevű könyvtár legenerálja az xml, valamint a Javascript változatot az aktuális beállításhoz képest.

Majd a Save gombra kattintva lementi az aktuális állását javascript, és az xml változatnak az Axios könyvtár segítségével.

A program támogatja kézi javascript hozzáadását is, ehhez a Blockly Block alatt található Text area mezőbe kell megírni a szükséges aktiváló kódott.



ábra 7 Program szerkesztő

És az alábbi szkript amit a program generált:

```
var device, mode, status2;

// Send data

if (true) {
  for (var count = 0; count < 10; count++) {
    send_data('1', 'relay', 'ON');
  }
}
```

Ezt feldolgozza a következő script ami az api.php fájlban található.

```
Route::middleware(['auth:sanctum', 'verified'])->post('update-program/{program_id}', function (Request $request, $program_id) {
    Programs::where('id', $program_id)->update(['name'=>$request->name, 'active'=>1, 'javascript_block'=>$request->javascript_block, 'xml_block'=>$request->xml_block]);
});
```

### 3.7. Programok futtatása:

A programok futtatása az ún run.vue fileban található meg. Ez gyakorlatilag egy Sandbox-ot hoz létre az aktuális futó szkriptnek, és ebben fut a leadott program is. Először is létrehozza az aktuális „futamot”, melynél elmenti a programot, a csapat id-t, a kezdeti dátumot, és NULL értéket ad a befejezési dátumnak (innen tudjuk, hogy aktív a futam).

A programot betöltés után a mounted függvény megnyitja és a Acorn fordító Sandbox-ába téve futtatja le. Szükséges megemlíteni, hogy a homokozóban 10 000 maximális lépés van meghatározva.

A programnak van egy belső konzolja amivel „kommunikál” a perifériákkal, és ezek segítségével állítja át az esp-k tulajdonságait, pl.

```
send_data("1","relay","1");
```

S ezt használja ki a Blockly-ba írt program is. Ezt a program elküldi az adott API csatlakozóra, majd lementi a MySQL mezőbe.

```
send_data(id,mode,status)
{
  axios.get('/api/device/js-api/'+id+'/'+mode+'/'+status);
},
```

Ezt a kérést fogadó szkript:.

```
Route::get('device/js-api/{id}/{mode}/{status}',function ($id,$mode,$status) {
    $device= Devices::findOrFail($id)->get();
    Devices::findOrFail($device[0]->id)->update(['mode'=>$mode,'status'=>$status]);
    return 'ok';
});
```

### 3.8. Kamerák kezelése:

Mint minden szabaduló szobában szükséges megfigyelő eszközöket elhelyezni a játékosok tippjeinek, vagy megakadásának megsegítésére. Ezért szükséges volt a programba beültetni egy Kamera kezelő modult, mely az aktuális szobára képes IP webkameráról képet adni.

Ezt úgy érjük el, hogy az esp hálózatba csatoljuk a webkamerákat is, és megadjuk azok streaming címét, ezáltal a program ha rákattintunk az adott kamerára képes annak képét megjeleníteni.

```

<template>
  <app-layout>
    <template #header>
      <h2 class="font-semibold text-xl text-gray-800 leading-tight">
        Cameras
      </h2>
    </template>

    <div class="py-12">
      <div class="max-w-7xl mx-auto sm:px-6 lg:px-8">
        <div class="bg-white overflow-hidden shadow-xl sm:rounded-lg">
          <div>
            <div class="bg-white shadow-md rounded my-6">
              <table class="min-w-max w-full table-auto">
                <thead>
                  <tr>
                    <th class="w-1/4 ...">#</th>
                    <th class="w-1/3 ...">Name:</th>
                    <th class="w-1/3 ...">Camera Show</th>
                  </tr>
                </thead>
                <tbody>
                  <tr v-for="row in cameras" :key="row.id">
                    <td>{{ row.id }}</td>
                    <td>{{ row.name }}</td>
                    <td>
                      <jet-button v-on:click="open_camera(row.id)">Show camera
                        <BIconArrowRightSquareFill class=""/>
                      </jet-button>
                      <span v-on:click="delete"><i class="bi bi-trash"></i> </span></td>
                    </td>
                  </tr>
                </tbody>
                <tfoot>
                  <tr>
                    <td class="w-1/3 ..."></td>
                    <td class="w-1/3 ..."></td>
                    <td class="w-1/5 ..."></td>
                  </tr>
                </tfoot>
              </table>
            </div>
          </div>
        </div>
      </div>
    </app-layout>
  </template>

  <script>
    import {Inertia} from '@inertiajs/inertia'
    import axios from 'axios';
    import AppLayout from '@/Layouts/AppLayout'
    import Welcome from '@/Jetstream/Welcome'
    import JetLabel from '@/Jetstream/Label'
    import JetButton from '@/Jetstream/Button'
    import JetInput from '@/Jetstream/Input'
    import {BIconArrowRightSquareFill} from 'bootstrap-icons-vue';
    import Button from "../Jetstream/Button";
  </script>

```



```

export default {
  props: {
    cameras: {
      type: Array,
      required: true
    }
  },
  data() {
    return {
      add_name: ""
    }
  },
  mounted: () => {
    //console.log(rows);
  },
  components: {
    Button,
    AppLayout,
    JetButton,
    JetInput,
    JetLabel,
    BIconArrowRightSquareFill
  },
  methods: {
    open_camera(id) {
      let tmp = 0, l = 0;

      this.cameras.forEach(element => {
        if (element.id == id) tmp = this.cameras[l];
        l++;
      });

      console.log(tmp);
      let url = tmp.url;
      console.log("Opening popup");
      let new_window = window.open("about:blank", tmp.name,
'width=300,height=300');
      new_window.document.write("<img src=\"\" + url + \"\" alt=\"Camera\"
/>");
    }
  }
}
</script>

```

### 3.9. Az eszközök kezelése:

Az eszközök kezelését két fő részre lehet bontani, van az automatikus kezelés (program általi), és van kézi vezérlés.

Az alábbi részben a kézi vezérlésről lesz szó.

A kézi, valamint az automata kezelés ugyan arra az API-ra hivatkozik.

```

Route::get('device/js-api/{id}/{mode}/{status}', function ($id, $mode, $status) {
  $device= Devices::findOrFail($id)->get();

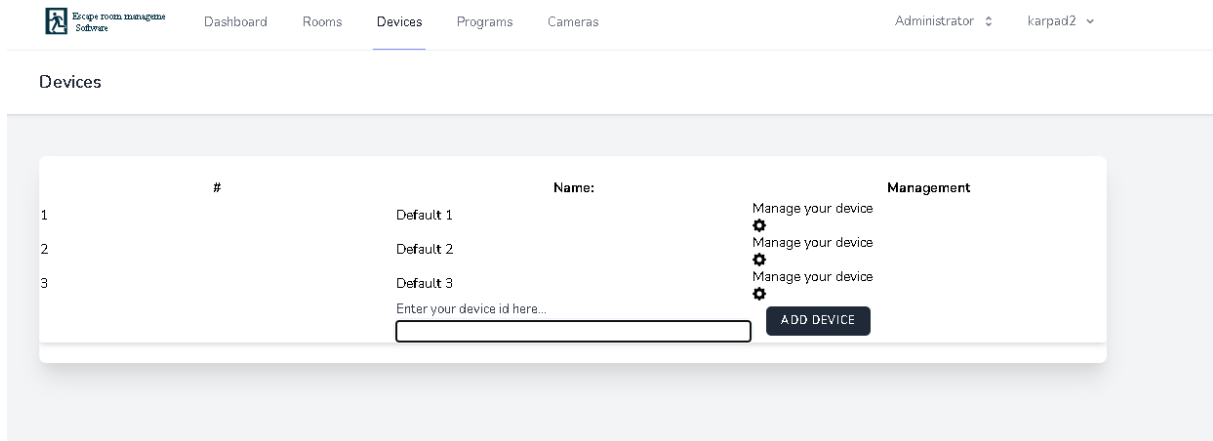
```

```

    Devices::findOrFail($device[0]->id)->update(['mode'=>$mode, 'status'=>$status]);
    return 'ok';
});

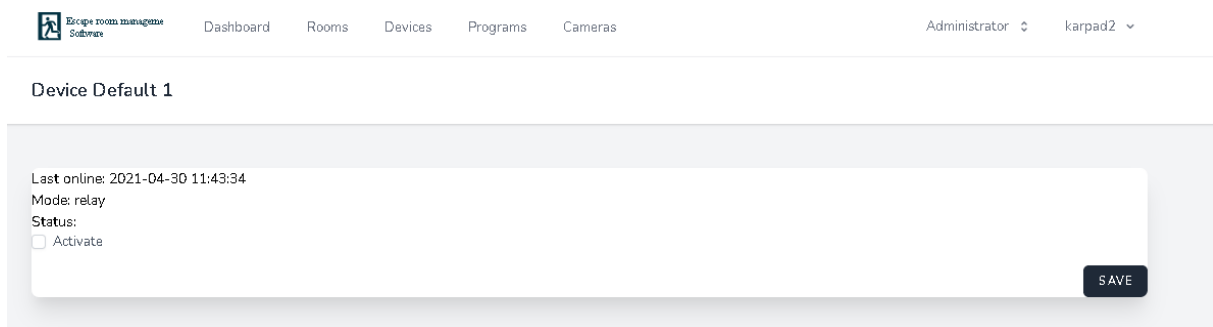
```

A vue rész ami kezeli az eszközöket:



ábra 8 Saját eszközök áttekintése

Valamint van lehetőség az eszközöknek a saját konfigurációjára is.



ábra 9 Az eszköz manuális beállítása

### 3.10. Az ESP32 programja:

Az ESP32 először is egy bejelentkezési fázison esik át, ahol egy szózott és az eszköz belső id-jének hash-ével jelentkezik be a kezelő API-ra.

### 3.11. A regisztráció menete:

Először is az ESP32 használja a SPIFFS tárolót, és itt be van állítva egy config.json file.

Ebből a fileból lehet beállítani az eszköznek a saját „szerver-ét”, de mivel ez egy központosított rendszer, ezért több eszközről is betud jelentkezni.

Amennyiben az eszköz friss, még nincs az esp32pass -nak értéke, akkor meglátogatja a szerver autentikációs oldalát.

```

void http_mod()
{
    String res="";
    Serial.print("[HTTP] begin...\n");
    String tmpaddress="";
    if(esp32_pass=="")
    {
        tmpaddress=web_server+"/api/device/add-device";
        Serial.println(tmpaddress);
        res=http_sys(tmpaddress);
        tmp1=JSON.parse(res);
        String tmp2="";
        tmp2=tmp1["password"];
        //Serial.println(JSON.stringify(tmp1));
        configobj["esp32pass"]=tmp2;
        Serial.println(JSON.stringify(configobj));
        save_config();
        ESP.restart();
    }
}

```

Ezt a rendszer fogadja, majd hozzáadja a devices táblához. Majd ezt az eszközt ezzel az ESP32pass lehet hozzáadni a felhasználóhoz.

Ezután a rendszer válaszbán megadja az ESP32 pass-t, lementi a config file-t.

```

Route::get('device/add-device', function (Request $request) {
    $id=Devices::create(['name'=>'Default', 'ip_address'=>$request->
getClientIp(), 'mode'=>'relay',
'status'=>'0', 'last_online'=>now(), 'room_id'=>1, 'pass-
word'=>'asdfghjkl'])->id;
    Devices::findOrFail($id)->update(['pass-
word'=>md5('salt'.$id.'salt')]);
    return Devices::findOrFail($id);
});

```

Az eszközt újra indítjuk.

### 3.12. Az eszköz bejelentkezése:

Mint az előbb említettem a rendszernek a bejelentkezéshez szükséges a jelszó használata.

Az eszköznek 4 féle módja van előre letárolva, de lehet, hogy ez a lista még később bővül.

- Relé üzemmód.
- RFID olvasó üzemmód.
- Bemeneti üzemmód.
- OLED kijelző üzemmód.

Ebben a megvalósításban az eszköz egyszerre csak egy módszert támogat, még pedig azt amit a hálózatról kiolvas.

Először is lekérdezzük a tárolt jelszóval az eszköz állapotát.

```
tmpaddress=web_server+"/api/device/status/"+esp32_pass;
res=http_sys(tmpaddress);
tmp1=JSON.parse(res);
String tmp3="";
tmp3=tmp1["status"];
tmp3=tmp1["mode"];
```

A php szkript ami kezeli a lekérdezést:

```
Route::get('device/status/{device_password}', function ($device_password)
{
    $device = Devices::where('password', $device_password)->get();
    Devices::findOrFail($device[0]->id)->
update(['last_online' => now()]);
    return $device[0];
});
```

Ha az eszköz a jó hash értékkel jelentkezik be, akkor az eszköz a modelljének megfelelő választ fogja kiírni, valamint a last\_online értéket is frissíti az aktuális idő pontra.

Ha a last\_online értéke és az aktuális idő különbsége nagyobb mint 5 perc, akkor a rendszer hibásnak minősül.

Ha a válasz nem egyezik az előre letárolt változattal, a mikrovezérlő újraindul.

```
if(espstatus!=tmp3)
{
    configobj["mode"]=tmp3;
    save_config();
    ESP.restart();
}
```

### 3.13. Relé üzemmód:

Ammenyiben a bemenetre a relé üzemmód van kapcsolva, akkor az eszköz elkezd a változóknak az elemeit értelmezi az értékét és egyet villant a led-en.

```
if(tmp3=="relay")
{
    tmp3=tmp1["status"];
    Serial.print("status:");
    tmp3=tmp1["status"];
    int stat=tmp1["status"];
    Serial.println(tmp3);

    if(stat==1)
    {
        digitalWrite(relaypin, 1);
    }
}
```

```

    blinking(1);
    Serial.println("Relay ON");
}
else
{
    digitalWrite(relaypin, 0);
    Serial.println("Relay OFF");
}
}

```

### 3.14. Bemenet üzemmód:

A bemeneti üzemmódban a kijelölt láb (25) és a GND (föld) közötti rövidre zárásként reagál. Ebben az üzemmódban egy kapcsoló reagálásra, pl. Reed relé összekapcsolására reagál.

```

else if (tmp3=="input")
{
    attachInterrupt(interruptpin, isr, FALLING);
}

```

S a program reagál erre a lábra és bejelentkezik a szerverre.

```

void IRAM_ATTR isr() {
    tmpaddress =web_server+"/api/device/input/"+esp32_pass+"/" +1;
    String res=http_sys(tmpaddress);
    blinking(3);
    tmp1=JSON.parse(res);
}

```

Ezt fogadó php szkript:

```

Route::get('device/input/{device_password}/{status}',
function ($device_password, $status) {
    $device = Devices::where('password', $device_password)->
update(['status' => $status]);
    Devices::findOrFail($device[0]->id)->update(['last_online' =>
now()]);
    return $device[0];
});

```

### 3.15. Megjelenítő üzemmód

A megjelenítő üzemmód lehető teszi, hogy egy OLED kijelzőre lehetséges legyen szöveget kiíratni.

Megjelenítése az ESP32:

```

else if (tmp3=="display")
{
    tmp3=tmp1["text"];
    if(display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {

```

```

display.clearDisplay();
display.setTextSize(1);           // Normal 1:1 pixel scale
display.setTextColor(WHITE);      // Draw white text
display.setCursor(0,0);           // Start at top-left corner
display.println(F(tmp3));
}
}

```

Fogadó szkript mely kezeli a megjelenítést:

```

Route::get('device/store/dev-api/{id}/{status}', function ($id, $text) {
    $device = Devices::where('id', $id)->update(['text' => $text]);

    return 'ok';
});

```

### 3.16. RFID üzemmód

A szoftver képes RFID kártya olvasására és jelezni a szoftver számára.

Az esp32 szkript:

```

else if(tmp3=="rfid")
{
    if ( ! rfid.PICC_IsNewCardPresent()) return;
    if ( ! rfid.PICC_ReadCardSerial()) return;

    Serial.print(F("PICC type: "));
    MFRC522::PICC_Type piccType = rfid.PICC_GetType(rfid.uid.sak);
    Serial.println(rfid.PICC_GetTypeName(piccType));
    if (piccType != MFRC522::PICC_TYPE_MIFARE_MINI &&
        piccType != MFRC522::PICC_TYPE_MIFARE_1K &&
        piccType != MFRC522::PICC_TYPE_MIFARE_4K) {
        Serial.println(F("Your tag is not of type MIFARE Classic."));
        return;
    }
    for (byte i = 0; i < 4; i++) nuidPICC[i] = rfid.uid.uidByte[i];
    Serial.println(F("The NUID tag is:"));
    Serial.print(F("In hex: "));
    s_rfid="";
    for(byte i=0;i<rfid.uid.size;i++)
    {
        s_rfid+=String(rfid.uid.uidByte[i],HEX);
    }
    tmpaddress=web_server+"/api/device/rfid/"+esp32_pass+"/"+s_rfid;
    res=http_sys(tmpaddress);
}

else
{
}

```

```
}
}
```

A szerver felőli kezelő felület:

```
Route::get('device/rfid/{device_password}/{code}',
function ($device_password, $code) {
    $device = Devices::where('password', $device_password)->update(['text' => $code]);
    Devices::findOrFail($device[0]->id)->update(['last_online' => now()]);
    return $device[0];
});
```

Mint látható az eszköz elküldi a kártya azonosító számát és beírja az adatbázisnak a text mezőjébe az értéket.

### 3.17. Raspberry PI beállítása:.

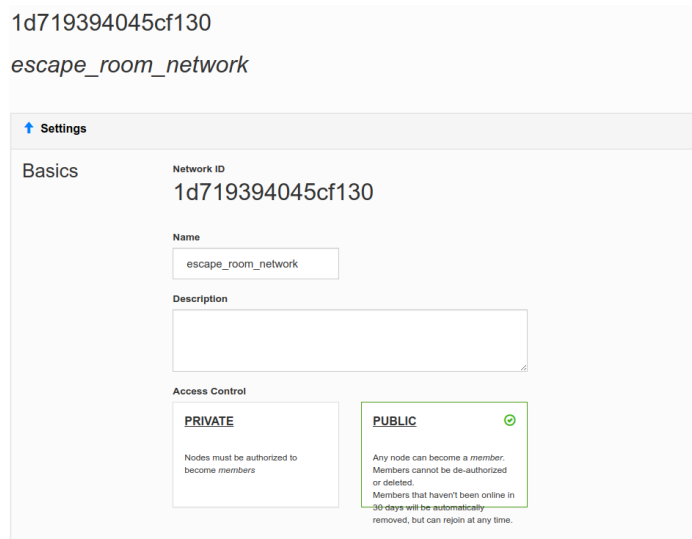
Zerotier alkalmazást feltelepítve beállítottam, hogy induláskor először lecsatlakozon a hálózatról, majd visszacsatlakozon azért, mert a kép fájl ami a github tárolóban fent van, ne jelentsen ütközést a legelső indulásnál.

```
zerotier-cli leave 1d719394045cf130
zerotier-cli join 1d719394045cf130
```

Rasp Ap nyílt hozzáférésű szoftvert felraktam Manjaro operációs rendszerre. Majd beállítottam, hogy a Zerotier hálózatra adjon hozzáférést a beállított SSID, és wifi jelszó segítségével.

```
SSID: Escape Room
Jelszó: escape_room
```

Majd felállítottam egy Webmin szolgáltatást is a rendszeren, ahol átírányítottam a kéréseket a VPN szolgáltatásra.



ábra 10 Zerotier kezelő felülete

### 3.18. A szoftver futtatása Raspberry PI-n:

Először is szükséges leklónozni a Github Repository tárolót.

```
git clone https://github.com/karpad2/obudai_diplomamunka
```

Ennek végeztével szükséges felrakni, manjaro operációs rendszerre a yay nevezetű szoftvert.

```
sudo pacman -Sy yay
```

Majd ha ez a program feltelepült a következő szkriptet futtatjuk le:

```
yay -S node
yay -S composer
```

Az /etc/php.ini file-ban engedélyezzük az iconv, valamint a pdo\_mysql kiegészítőket.

Belépünk a repositoryba, majd az esc\_room\_server mappába.

```
composer update
```

A szükséges php függőségeket feltelepítjük, majd elkészítjük a Javascript beli függőséggeket is.

```
npm install
npm run prod
```

Majd elkészítjük az optimalizált verziót a futatásra.

Majd elindítjuk a következő parancsal a programot:

```
sudo php artisan serve --host 0.0.0.0 --port 80
```



## 4. Összefoglalás

Úgy vélem, hogy a szakdolgozatban elért célokat teljesítettük. Egy olyan rendszert sikerül megvalósítani, amely hardveres modulok mellett tartalmaz egy klienst és egy web alkalmazást. A kifejlesztett modulok tökéletesen működnek.

Az esp32 és a webalkalmazás közötti kommunikáció megfelelő működést biztosít. A vezérlés kiegészítését jelentő, a Raspberry-n futó szerver és a web alkalmazás is olyan lehetőségekkel bővült, amelyeket a fejlesztés elején kitűztem:

- a vezérlés mobil eszközről is megvalósulhat;
- Programokat lehet készíteni, valamint futtatni;
- Az eszközök állapota nyomon követhető a menüben is;

A rendszer úgy lett kialakítva, hogy a jövőben bármikor tovább lehessen fejleszteni. Erre a felhasználók javaslatokat is tehetnek a Githubon[5].

A alkalmazást lehetne tovább fejleszteni Dockerba, de mivel a Docker ARM támogatottsága még eléggé limitált, ezért ezt a célt még nem sikerült megvalósítani, tehát egy „kattintással elindítani”.

## **5. Idegen nyelvű tartalmi összefoglaló**

Escape room management software which is based on ESP32, contains admin user interface, and multiple features for easier setup, reset, and for measuring time.

## 6. Irodalomjegyzék

- [1] Szabaduló szoba [https://hu.wikipedia.org/wiki/Szabaduló\\_játék](https://hu.wikipedia.org/wiki/Szabaduló_játék)
- [2] <https://nodejs.org>, Node.JS alapok
- [3] [www.inf.u-szeged.hu/miszak/utmutato](http://www.inf.u-szeged.hu/miszak/utmutato)- Arduino – kezdő lépések
- [4] <https://modi-j.webnode.hu/> - A Raspberry Pi számítógép
- [5] Git tároló [https://github.com/karpad2/obudai\\_diplomamunka](https://github.com/karpad2/obudai_diplomamunka)
- [6] A program logója  
<https://www.lastingimpressionsonline.co.uk/image/cache/catalog/Product%20Images/Section%203/BS%20Fire%20Exit/03205-500x500.png>

## **7. Mellékletek**

**Nyomtatott melléklet**

**Elektronikus melléklet**

CD, mely tartalmazza a dokumentációt és összes forráskódot.