

**ÓBUDAI EGYETEM**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Bánki Donát Gépész és Biztonságtechnikai Mérnöki Kar**

**Szabaduló szoba vezérlő szoftver tervezése**

**OE-BGK** Hallgató neve: **Kovács Árpád**

**2021.** Neptun kód: BPJZ56

**Tartalomjegyzék**

[1.](#_30j0zll) Bevezetés 4

[2.](#_1fob9te) Szakirodalom elemző feldolgozása 5

[2.1](#_3znysh7) Hivatkozások 5

[2.1.1](#_2et92p0) Hivatkozások sorrendje 6

[3.](#_tyjcwt) Saját munka 7

[3.1](#_3dy6vkm) Rajzok, képek, diagramok, ábrák 7

[3.2](#_1t3h5sf) Képletek 8

[3.3](#_4d34og8) Felsorolás 10

[4.](#_2s8eyo1) Összefoglalás 12

[5.](#_17dp8vu) Idegen nyelvű tartalmi összefoglaló 13

[6.](#_3rdcrjn) Felhasznált irodalom vagy Irodalomjegyzék 14

[7.](#_26in1rg) Mellékletek 15

# Bevezetés

A feladat egy olyan szabaduló szoba vezérlés létrehozása, mellyel akár a programot távolról is lehessen futattni. 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 segítését, a csapatok nyomon követését, és utoljára de nem utolsósorban a szobák programozását.

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

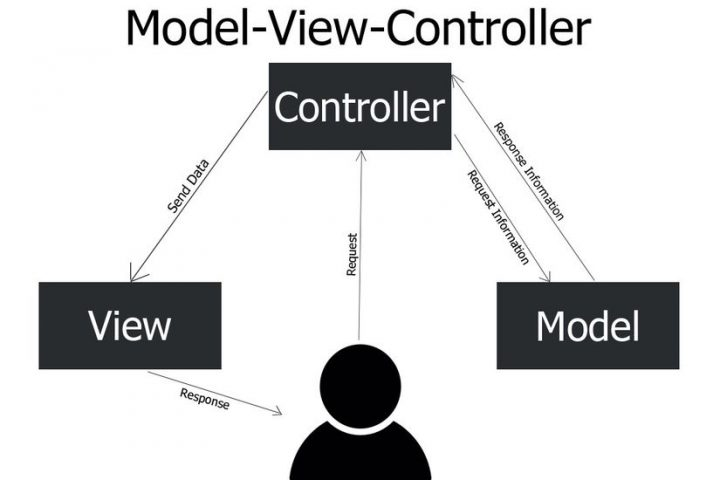
# Szakirodalom elemző feldolgozása

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 mivel a naptári év immáron 2021 úgy gondoltam, hogy ez túlságosan abszurd lenne ezért választottam a Laravel nevezetű PHP keretrendszert, mivel biztonságosabb.

A Laravel keretrendszer egy Symfony támogatottságra épülő szoftver mely imáron 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 MVC modell

Ezzek végeztével elkezdtem kidolgozni a feladathoz szükséges táblázati elemeket, később feltelepítettem a Laravel Jetstram 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 létrehozzását. Mely csak Ajax használatával kommunikál a rendszerrel, a többit egy előre fordított Javascript-ten keresztül tölt 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 tartozzó weboldalakat létrehozzni. Ezek egy Vue.js könyvtárban találhatóak, melyeknek direkt kommunikáció-val rendelkeznek a weboldalhoz. Ezt egy úgy nevezett 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.

A modellek

A modellek amik tárolják az adatokat, amelyek segítségével kommunikálunk az adatbázzissal, valamint ezeket az adatokat írjuk bele.

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

View vagyis 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.

Controller vagyis a 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.

Babel fordító könyvtár

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

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 csak 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 konzol létrehozni mely lehetővé teszi a felhasználóknak a blockly által támogatott blockok futattását, valamint azok reakcióját a valós rendszerrel.

Raspberry PI 400

A szolgáltatás futattására én a Raspberry PI 400-as mikroszámítógépet használtam fel. Ezt használja a felhasználó az eszköz kezelésére.

Manjaro ARM

Manjaro ARM változatta ez az operációs rendszer fut a Raspberry PI 400-on. A Manjaro egy Arch típusú Linux disztrubició.

MySQL

A MySQL egy relációs adatbázis kezelő szoftver.

PHP

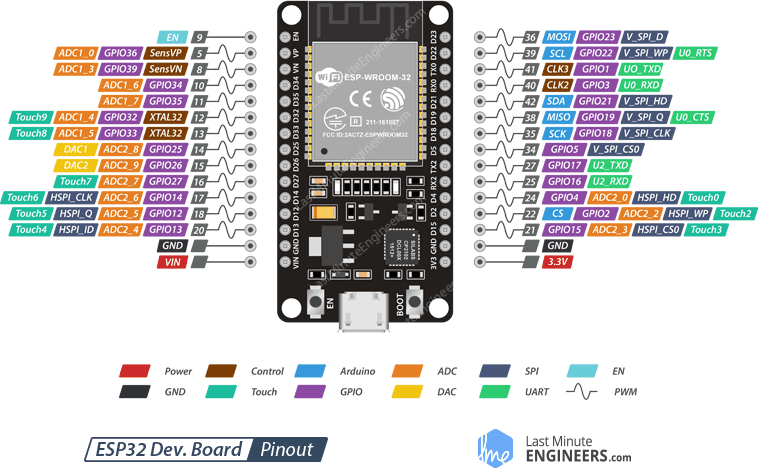
PHP vagyis prehypertext szkript nyelv ebben íródott a Laravel rendszer.

NPM/Node.JS

A rendszer az npm könyvártkezelő segítségevel építí fel magát.

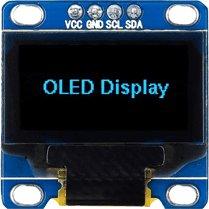
ESP32

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 ESP32

Az ESP32 támogat több -féle protokkolt egyenlőre 3 dolog lett kidolgozva képes relé üzemmódban dolgozni, érzékeli azt és elküldi az RFID kártya ID-jét a szervernek, valamint képes bemenetti módban is dolgozni azáltal, hogy az megjelölt () portott rövidre zárjuk a földdel. Utoljára és nem utolsósorban képes az eszköz megjelenítő eszközként is dolgozni amennyiben egy oled kijelzőtt kötünk rá.



ábra OLED kijelző

Zerotier szolgáltatás

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

# Saját munka

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

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, mely egy lapos webalkalmazást lehet vele létrehozni.

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

Majd folytatva elkezdtem a modelleket létrehozni.

|  |
| --- |
| 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.

## 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();  }); |

Ez által létrehoztuk a struktúrát.

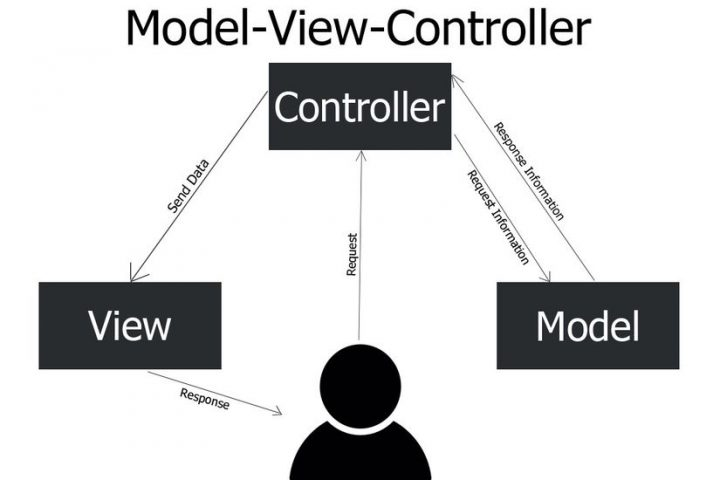
A következő paranccsal meg lementjük a struktúrát az adatbázisba.

|  |
| --- |
| php artisan migrate |

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

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

MVC modell:.



Ez szerint elkezdtem kidolgozni az MVC modellt. Megkell említeni, hogy a rendszer elsőlegesen AJAX adatkommunikációt használ, míg a programkínézetek előre vannak fordítva az app.js file-ba, a Babel könyvtár segítségével.

Modellekel 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',  '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 moddelje:

|  |
| --- |
| 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 könyvtárát és az ahhoz tartozó tábla kezelő modulokat is.

Ezt a modult feltelepítve létrehoztam a következő rendszert.

Az Api.php route-k amivel, főleg az esp-k útvonalai találhatóak file tartalma.

|  |
| --- |
| <?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'=>'asdfghjkl'])->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 (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]);   }); |

Hálózatkezelés:

Az alapötlete a szervernek és a használatának a peertől-peerig való használat. Ez magába foglalja az adatok védelmét is mivel az adatok 80-as porton utaznak ezért szükséges volt védelmet használni ezért szükséges volt bevezeti a Raspberry Pi-t mint az esp32 hálózatnak a „router-ét” amivel eléri a szervert, ehhez én a Zerotier nevezetű nyílt forrású VPN szolgáltatás vettem fel, melyel 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 szintén hálózati hídat 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 nekülön bázissokkal dolgozzak.

A nézetek:

## PHP rész:.

A laravel alapértelmezetten Vue.JS keretrendszert használja. Ezért az útvonalakat is Vue, valamint kinézeti file-okat. Vue-ban kelett 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){  Cameras::*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){  Cameras::*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'); |

## 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: {</script>  },  data: ()=>  {  Itt találhatóak program változói,  },  methods:  {  program()  },  mounted: mounted()  {  Meghíváskor lefutó szkript  }  }  <style>  Míg a stilusban 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 amíg meghatározzák ezek az alábbi fileoknak a müködését, 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' }); |

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>  </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 változatát, és a javascript változatát programnak ezt a Blockly felhasználja és a Vue beilleszti a két kijelölt textarea html tag-be. Minden egyes változatatásnál a felületen a Blockly.JS nevű könyvtár legenerálja az xml változatott valamint a Javascript változatott 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.

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]);   }); |

Programok futattása:

A programok futattása az ún run.vue fileban található meg. Ez gyakorlatilag egy Sandbox-ot hozz 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'; }); |

Raspberry PI beállítása:.

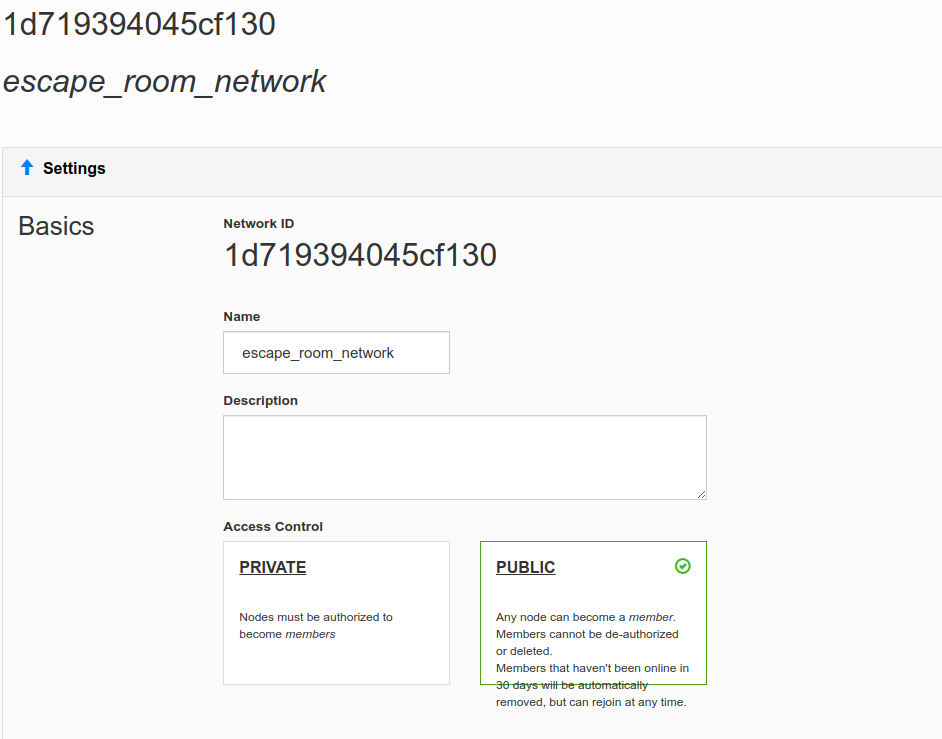
Zerotier alkalmazást feltelepítve beállítottam, hogy induláskor először lecsatlakozon a hálózatról, majd visszacsatlakozon azért, mert a képfájl ami a github tárolóban fent, van ne jelentsen ütközést, a legelső indulásnál.

|  |
| --- |
| zerotier-cli leave 1d719394045cf130  zerotier-cli join 1d719394045cf130 |

Rasp Ap nyílt hozzáférésű szoftver 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 kéréseket a VPN szolgáltatásra.

A konfiguráció

ábra Zerotier kezelő felülete

|  |
| --- |
|  |

## Összefoglalás

A dolgozat utolsó része a rövid tartalmi összefoglaló, mely minimum egy oldal legyen, terjedelemben az egész Szakdolgozat maximum 5%-a lehet (1500-2500 karakter). Tartalmazza a kitűzött célok, felvetett problémák megoldását továbbá a következtetéseket. Térjen ki a gyakorlati alkalmazhatóságra.

# 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.

# Felhasznált irodalom vagy Irodalomjegyzék

Példák az egyes publikációk megnevezésére.

[4] Miko, B.: Hybrid inteligence system for mold cost estination. VI. International tools conference (ITC) 2007. Zlin, May 22-23, 2007. CD. ISBN-978-80-7318-572-5

Internetes publikáció esetében: Szerző (vagy Szerkesztő): Cím. Hely. Évszám. Honlap elérhetősége.

[5] Dr. Szabó László: Forgácsolás, hegesztés. Miskolc 2000.

<http://mek.oszk.hu/01200/01200/html/html>

# Mellékletek

Minden mellékletet sorszámmal, megnevezéssel kell ellátni. A szakdolgozat szövegi részében hivatkozni kell (1. sz. melléklet).

1. sz. melléklet:

2. sz. melléklet:

https://medium.com/@fmacedoo/standalone-application-with-electron-react-and-sqlite-stack-9536a8b5a7b9

https://lastminuteengineers.com/oled-display-esp32-tutorial/

<https://lastminuteengineers.com/multiple-ds18b20-esp32-web-server-tutorial/>

<https://medium.com/@dannyhuang_75970/learning-laravel-controllers-101-ad28d2bb5569>

https://github.com/RaspAP/raspap-webgui

https://medium.com/swlh/markup-images-with-vue-js-a4190d302005