**Project PCBuilder**

Backend Fejlesztői Dokumentáció

Ez a backend részleg a Laravel keretrendszer felhasználásával készült, PHP nyelven íródott. A Laravel verziója a 10-es.

**Backend Telepítési Útmutató**

1. **PHP Telepítése**: Ellenőrizd, hogy a számítógépeden telepítve van-e a PHP a következő paranccsal:   
   “php -v”

Ha nincs telepítve, telepítsd a PHP-t a [hivatalos weboldalról](https://www.php.net/).

1. **Composer Telepítése**: A Composer egy PHP függőségkezelő, amelyet a Laravel projektekhez használnak. Telepítsd a Composer-t a [hivatalos weboldalról](https://getcomposer.org/).
2. **Laravel Telepítése**: Telepítsd a Laravel-t a Composer segítségével a következő parancs segítségével:  
   “composer global require laravel/installer”
3. **Projekt Klónozása**: Klónozd le a projektet a GitHub-ról vagy más tárolórendszerből:  
   “git clone <https://github.com/Phreelosu/502tackshooter.git>”
4. **Projekt Függőségeinek Telepítése**: Navigálj a projekt könyvtárába, majd telepítsd a függőségeket a Composer segítségével:  
   “composer install”
5. **.env Állomány Beállítása**: Másold át a `**.env.example**` állományt `**.env**`-re (ha már van `**.env**` akkor írd felül), majd állítsd be a környezeti változókat, például az adatbázis kapcsolódási információkat.
6. **Adatbázis Migrációk és Seeder-ek Futtatása**: Futtasd a migrációkat az adatbázis táblák létrehozásához:  
   “php artisan migrate:fresh”   
   “php artisan db:seed”
7. **Szerver Indítása**: Indítsd el a Laravel beépített fejlesztői szervert a következő paranccsal:  
   “php artisan serve”  
     
   A projekt most már fut a saját helyi fejlesztői környezetedben, és elérhető a `**http://localhost:8000`** címen.

**Controller-ek**

Az **AlertController** felelős az riasztások kezeléséért és e-mail értesítések küldéséért a felhasználóknak.

Metódusok:

* **sendEmail($user, $time): void**
  + **Paraméterek**:
    - **$user**: Az e-mail értesítés címzettje.
    - **$time**: Az értesítés időpontja.
  + **Leírás**: Ez a metódus e-mail értesítést küld a megadott felhasználónak. Elkészíti az e-mail tartalmát, majd a Laravel Mail szolgáltatás segítségével elküldi az e-mailt. Ez sajnos jelenlegi állapotában **NEM** funkcionális.

A **BanController** felelős a felhasználók letiltási adatok kezeléséért.

Metódusok:

* **getLoginAttempts($email: string): int**
  + **Paraméter:**
    - **$email**: A felhasználó e-mail címe.
  + **Visszatérési érték:**
    - Az aktuális bejelentkezési kísérletek száma.
  + **Leírás:** Ez a metódus visszaadja az adott e-mail címhez kapcsolódó felhasználó bejelentkezési kísérleteinek számát.
* **setLoginAttempts($email: string): void**
  + **Paraméter:**
    - **$email**: A felhasználó e-mail címe.
  + **Leírás:** Ez a metódus növeli az adott e-mail címhez tartozó felhasználó bejelentkezési kísérleteinek számát.
* **getBannedTime($email: string): Carbon|null**
  + **Paraméter:**
    - **$email**: A felhasználó e-mail címe.
  + **Visszatérési érték:**
    - Az időpont, amikor a felhasználó letiltva van, vagy **null**, ha nincs letiltva.
  + **Leírás:** Ez a metódus visszaadja az adott e-mail címhez kapcsolódó felhasználó letiltási idejét.
* **setBannedTime($email: string): void**
  + **Paraméter:**
    - **$email**: A felhasználó e-mail címe.
  + **Leírás:** Ez a metódus beállítja az adott e-mail címhez tartozó felhasználó letiltási idejét. A felhasználót 60 másodpercig letiltja az aktuális időponttól számítva.
* **resetBannedData($email: string): void**
  + **Paraméter:**
    - **$email**: A felhasználó e-mail címe.
  + **Leírás:** Ez a metódus visszaállítja az adott e-mail címhez kapcsolódó felhasználó bejelentkezési kísérleteinek számát és letiltási idejét.

**Sajnos a kódot leszámítva a felhasználót hiába tiltja ki 3 sikertelen bejelentkezés után, ettől függetlenül be tud jelentkezni 4. alkalommal sikeresen.**

A **CaseTypeController** felelős a gépház típusok kezeléséért.

Metódusok:

* **index(): JSON**
  + **Visszatérési érték:**
    - Egy JSON válasz, amely tartalmazza az összes gépház típust.
  + **Leírás:** Ez a metódus visszaadja az összes gépház típust.
* **show($id: int): JSON**
  + **Paraméter:**
    - **$id**: Az gépház típus azonosítója.
  + **Visszatérési érték:**
    - Egy JSON válasz, amely tartalmazza az adott azonosítójú gépház típust.
  + **Leírás:** Ez a metódus visszaadja az adott azonosítójú gépház típust.

A **ColorController** felelős a színek kezeléséért.

Metódusok:

* **index(): JSON**
  + **Visszatérési érték:**
    - Egy JSON válasz, amely tartalmazza az összes színt.
  + **Leírás:** Ez a metódus visszaadja az összes színt.
* **store($request: Request): JSON**
  + **Paraméter:**
    - **$request**: Az új szín létrehozásához szükséges kérés.
  + **Visszatérési érték:**
    - Egy JSON válasz, amely tartalmazza az újonnan létrehozott színt.
  + **Leírás:** Ez a metódus létrehozza az új színt.
* **show($id: int): JSON**
  + **Paraméter:**
    - **$id**: A szín azonosítója.
  + **Visszatérési érték:**
    - Egy JSON válasz, amely tartalmazza az adott azonosítójú színt.
  + **Leírás:** Ez a metódus visszaadja az adott azonosítójú színt.
* **update($request: Request, $id: int): JSON**
  + **Paraméter:**
    - **$request**: A frissítendő színhez szükséges kérés.
    - **$id**: A frissítendő szín azonosítója.
  + **Visszatérési érték:**
    - Egy JSON válasz, amely tartalmazza a frissített színt.
  + **Leírás:** Ez a metódus frissíti az adott azonosítójú színt.
* **destroy($id: int): JSON**
  + **Paraméter:**
    - **$id**: A törlendő szín azonosítója.
  + **Visszatérési érték:**
    - Egy JSON válasz null értékkel.
  + **Leírás:** Ez a metódus törli az adott azonosítójú színt.

A **ConfigController** felelős a PC konfigurációk kezeléséért.

Metódusok:

* **newConfig($request: Request): JSON**
  + **Paraméter:**
    - **$request**: Az új konfiguráció létrehozásához szükséges kérés.
  + **Visszatérési érték:**
    - Egy JSON válasz, amely jelzi az újonnan létrehozott konfiguráció létrehozásának sikerét.
  + **Leírás:** Ez a metódus létrehozza az új konfigurációt a kapott adatok alapján.
* **modifyConfig($request: Request): JSON**
  + **Paraméter:**
    - **$request**: A módosítandó konfigurációhoz szükséges kérés.
  + **Visszatérési érték:**
    - Egy JSON válasz, amely jelzi a konfiguráció módosításának sikerét.
  + **Leírás:** Ez a metódus módosítja a megadott konfigurációt a kapott adatok alapján.
* **deleteConfig($request: Request): JSON**
  + **Paraméter:**
    - **$request**: A törlendő konfigurációhoz szükséges kérés.
  + **Visszatérési érték:**
    - Egy JSON válasz, amely jelzi a konfiguráció törlésének sikerét.
  + **Leírás:** Ez a metódus törli a megadott konfigurációt.
* **getConfig($id: int): JSON**
  + **Paraméter:**
    - **$id**: A lekérni kívánt konfiguráció azonosítója.
  + **Visszatérési érték:**
    - Egy JSON válasz, amely tartalmazza a lekért konfigurációt.
  + **Leírás:** Ez a metódus visszaadja a megadott azonosítójú konfigurációt.
* **getConfigs(): JSON**
  + **Visszatérési érték:**
    - Egy JSON válasz, amely tartalmazza az összes felhasználóhoz tartozó konfigurációt.
  + **Leírás:** Ez a metódus visszaadja az összes felhasználóhoz tartozó konfigurációkat.

A **CPUController** felelős a CPU-k kezeléséért.

Metódusok:

* **index(): JSON**
  + **Visszatérési érték:**
    - Egy JSON válasz, amely tartalmazza az összes CPU-t.
  + **Leírás:** Ez a metódus visszaadja az összes rendelkezésre álló CPU-t.
* **show($id: int): JSON**
  + **Paraméter:**
    - **$id**: A lekérni kívánt CPU azonosítója.
  + **Visszatérési érték:**
    - Egy JSON válasz, amely tartalmazza a lekért CPU-t.
  + **Leírás:** Ez a metódus visszaadja a megadott azonosítójú CPU-t.

A **CPUCoolerController** felelős a CPU hűtők kezeléséért.

Metódusok:

* **index(): JSON**
  + **Visszatérési érték:**
    - Egy JSON válasz, amely tartalmazza az összes CPU hűtőt.
  + **Leírás:** Ez a metódus visszaadja az összes rendelkezésre álló CPU hűtőt.
* **show($id: int): JSON**
  + **Paraméter:**
    - **$id**: A lekérni kívánt CPU hűtő azonosítója.
  + **Visszatérési érték:**
    - Egy JSON válasz, amely tartalmazza a lekért CPU hűtőt.
  + **Leírás:** Ez a metódus visszaadja a megadott azonosítójú CPU hűtőt.

A **GPUController** felelős a GPU-k kezeléséért.

Metódusok:

* **index(): JSON**
  + **Visszatérési érték:**
    - Egy JSON válasz, amely tartalmazza az összes GPU-t.
  + **Leírás:** Ez a metódus visszaadja az összes rendelkezésre álló GPU-t.
* **show($id: int): JSON**
  + **Paraméter:**
    - **$id**: A lekérni kívánt GPU azonosítója.
  + **Visszatérési érték:**
    - Egy JSON válasz, amely tartalmazza a lekért GPU-t.
  + **Leírás:** Ez a metódus visszaadja a megadott azonosítójú GPU-t.

A **GPUMemoryController** felelős a GPU memóriák kezeléséért.

Metódusok:

* **index(): JSON**
  + **Visszatérési érték:**
    - Egy JSON válasz, amely tartalmazza az összes GPU memóriát.
  + **Leírás:** Ez a metódus visszaadja az összes rendelkezésre álló GPU memóriát.
* **show($id: int): JSON**
  + **Paraméter:**
    - **$id**: A lekérni kívánt GPU memória azonosítója.
  + **Visszatérési érték:**
    - Egy JSON válasz, amely tartalmazza a lekért GPU memóriát.
  + **Leírás:** Ez a metódus visszaadja a megadott azonosítójú GPU memóriát.

Az **IHDCapacityController** felelős az belső merevlemezek (IHD) kapacitásainak kezeléséért.

Metódusok:

* **index(): JSON**
  + **Visszatérési érték:**
    - Egy JSON válasz, amely tartalmazza az összes belső merevlemez kapacitását.
  + **Leírás:** Ez a metódus visszaadja az összes rendelkezésre álló belső merevlemez kapacitását.
* **show($id: int): JSON**
  + **Paraméter:**
    - **$id**: A lekérni kívánt belső merevlemez kapacitás azonosítója.
  + **Visszatérési érték:**
    - Egy JSON válasz, amely tartalmazza a lekért belső merevlemez kapacitását.
  + **Leírás:** Ez a metódus visszaadja a megadott azonosítójú belső merevlemez kapacitását.

Az **IHDController** felelős az belső merevlemezek (IHD) kezeléséért.

Metódusok:

* **index(): JSON**
  + **Visszatérési érték:**
    - Egy JSON válasz, amely tartalmazza az összes rendelkezésre álló belső merevlemezt.
  + **Leírás:** Ez a metódus visszaadja az összes rendelkezésre álló belső merevlemezt.
* **show($id: int): JSON**
  + **Paraméter:**
    - **$id**: A lekérni kívánt belső merevlemez azonosítója.
  + **Visszatérési érték:**
    - Egy JSON válasz, amely tartalmazza a lekért belső merevlemezt.
  + **Leírás:** Ez a metódus visszaadja a megadott azonosítójú belső merevlemezt.

Az **IHDFormFactorController** felelős az IHD formátumfaktorok kezeléséért.

Metódusok:

* **index(): JSON**
  + **Visszatérési érték:**
    - Egy JSON válasz, amely tartalmazza az összes rendelkezésre álló IHD formátumfaktort.
  + **Leírás:** Ez a metódus visszaadja az összes rendelkezésre álló IHD formátumfaktort.
* **show($id: int): JSON**
  + **Paraméter:**
    - **$id**: A lekérni kívánt IHD formátumfaktor azonosítója.
  + **Visszatérési érték:**
    - Egy JSON válasz, amely tartalmazza a lekért IHD formátumfaktort.
  + **Leírás:** Ez a metódus visszaadja a megadott azonosítójú IHD formátumfaktort.

Az **IHDInterfaceController** felelős az IHD interfészek kezeléséért.

Metódusok:

* **index(): JSON**
  + **Visszatérési érték:**
    - Egy JSON válasz, amely tartalmazza az összes rendelkezésre álló IHD interfészt.
  + **Leírás:** Ez a metódus visszaadja az összes rendelkezésre álló IHD interfészt.
* **show($id: int): JSON**
  + **Paraméter:**
    - **$id**: A lekérni kívánt IHD interfész azonosítója.
  + **Visszatérési érték:**
    - Egy JSON válasz, amely tartalmazza a lekért IHD interfészt.
  + **Leírás:** Ez a metódus visszaadja a megadott azonosítójú IHD interfészt.

Az **IHDTypeController** felelős az IHD típusok kezeléséért.

Metódusok:

* **index(): JSON**
  + **Visszatérési érték:**
    - Egy JSON válasz, amely tartalmazza az összes rendelkezésre álló IHD típust.
  + **Leírás:** Ez a metódus visszaadja az összes rendelkezésre álló IHD típust.
* **show($id: int): JSON**
  + **Paraméter:**
    - **$id**: A lekérni kívánt IHD típus azonosítója.
  + **Visszatérési érték:**
    - Egy JSON válasz, amely tartalmazza a lekért IHD típust.
  + **Leírás:** Ez a metódus visszaadja a megadott azonosítójú IHD típust.

A **MemoryController** felelős a memóriák kezeléséért.

Metódusok:

* **index(): JSON**
  + **Visszatérési érték:**
    - Egy JSON válasz, amely tartalmazza az összes rendelkezésre álló memóriát.
  + **Leírás:** Ez a metódus visszaadja az összes rendelkezésre álló memóriát.
* **show($id: int): JSON**
  + **Paraméter:**
    - **$id**: A lekérni kívánt memória azonosítója.
  + **Visszatérési érték:**
    - Egy JSON válasz, amely tartalmazza a lekért memóriát.
  + **Leírás:** Ez a metódus visszaadja a megadott azonosítójú memóriát.

A **MemoryModulesController** felelős a memóriamodulok kezeléséért.

Metódusok:

* **index(): JSON**
  + **Visszatérési érték:**
    - Egy JSON válasz, amely tartalmazza az összes rendelkezésre álló memóriamodult.
  + **Leírás:** Ez a metódus visszaadja az összes rendelkezésre álló memóriamodult.
* **show($id: int): JSON**
  + **Paraméter:**
    - **$id**: A lekérni kívánt memóriamodul azonosítója.
  + **Visszatérési érték:**
    - Egy JSON válasz, amely tartalmazza a lekért memóriamodult.
  + **Leírás:** Ez a metódus visszaadja a megadott azonosítójú memóriamodult.

A **MOBOController** felelős az alaplapok kezeléséért.

Metódusok:

* **index(): JSON**
  + **Visszatérési érték:**
    - Egy JSON válasz, amely tartalmazza az összes rendelkezésre álló alaplapot.
  + **Leírás:** Ez a metódus visszaadja az összes rendelkezésre álló alaplapot.
* **show($id: int): JSON**
  + **Paraméter:**
    - **$id**: A lekérni kívánt alaplap azonosítója.
  + **Visszatérési érték:**
    - Egy JSON válasz, amely tartalmazza a lekért alaplapot.
  + **Leírás:** Ez a metódus visszaadja a megadott azonosítójú alaplapot.

A **MOBOFFController** felelős az alaplap formátumfaktorának kezeléséért.

Metódusok:

* **index(): JSON**
  + **Visszatérési érték:**
    - Egy JSON válasz, amely tartalmazza az összes rendelkezésre álló alaplap formátumfaktorát.
  + **Leírás:** Ez a metódus visszaadja az összes rendelkezésre álló alaplap formátumfaktorát.
* **show($id: int): JSON**
  + **Paraméter:**
    - **$id**: A lekérni kívánt alaplap formátumfaktorának azonosítója.
  + **Visszatérési érték:**
    - Egy JSON válasz, amely tartalmazza a lekért alaplap formátumfaktorát.
  + **Leírás:** Ez a metódus visszaadja a megadott azonosítójú alaplap formátumfaktorát.

A **MOBOMMController** felelős az alaplap maximális memóriájának kezeléséért.

Metódusok:

* **index(): JSON**
  + **Visszatérési érték:**
    - Egy JSON válasz, amely tartalmazza az összes rendelkezésre álló alaplap maximális memóriáját.
  + **Leírás:** Ez a metódus visszaadja az összes rendelkezésre álló alaplap maximális memóriáját.
* **show($id: int): JSON**
  + **Paraméter:**
    - **$id**: A lekérni kívánt alaplap maximális memóriájának azonosítója.
  + **Visszatérési érték:**
    - Egy JSON válasz, amely tartalmazza a lekért alaplap maximális memóriáját.
  + **Leírás:** Ez a metódus visszaadja a megadott azonosítójú alaplap maximális memóriáját.

A **MOBOMSController** felelős az alaplap memóriaslotjainak kezeléséért.

Metódusok:

* **index(): JSON**
  + **Visszatérési érték:**
    - Egy JSON válasz, amely tartalmazza az összes rendelkezésre álló alaplap memóriaslotjait.
  + **Leírás:** Ez a metódus visszaadja az összes rendelkezésre álló alaplap memóriaslotjait.
* **show($id: int): JSON**
  + **Paraméter:**
    - **$id**: A lekérni kívánt alaplap memóriaslotjainak azonosítója.
  + **Visszatérési érték:**
    - Egy JSON válasz, amely tartalmazza a lekért alaplap memóriaslotjait.
  + **Leírás:** Ez a metódus visszaadja a megadott azonosítójú alaplap memóriaslotjait.

A **PC\_CaseController** felelős a számítógépházak kezeléséért.

Metódusok:

* **index(): JSON**
  + **Visszatérési érték:**
    - Egy JSON válasz, amely tartalmazza az összes rendelkezésre álló számítógépházat.
  + **Leírás:** Ez a metódus visszaadja az összes rendelkezésre álló számítógépházat.
* **store(Request $request): JSON**
  + **Paraméterek:**
    - **$request**: A HTTP kérés objektuma, amely tartalmazza az új számítógépház adatait.
  + **Visszatérési érték:**
    - Egy JSON válasz az újonnan létrehozott számítógépházzal.
  + **Leírás:** Ez a metódus létrehozza és tárolja az új számítógépházat az adatbázisban.
* **show($id: int): JSON**
  + **Paraméter:**
    - **$id**: A lekérni kívánt számítógépház azonosítója.
  + **Visszatérési érték:**
    - Egy JSON válasz a lekért számítógépházzal.
  + **Leírás:** Ez a metódus visszaadja a megadott azonosítójú számítógépházat.
* **update(Request $request, $id: int): JSON**
  + **Paraméterek:**
    - **$request**: A HTTP kérés objektuma, amely tartalmazza a frissítendő számítógépház adatait.
    - **$id**: A frissítendő számítógépház azonosítója.
  + **Visszatérési érték:**
    - Egy JSON válasz a frissített számítógépházzal.
  + **Leírás:** Ez a metódus frissíti az adott azonosítójú számítógépház adatait az adatbázisban.
* **destroy($id: int): JSON**
  + **Paraméter:**
    - **$id**: A törlendő számítógépház azonosítója.
  + **Visszatérési érték:**
    - Egy JSON válasz, amely értesítést ad a sikeres törlésről.
  + **Leírás:** Ez a metódus törli az adott azonosítójú számítógépházat az adatbázisból.

A **PSUController** felelős a tápegységek kezeléséért.

Metódusok:

* **index(): JSON**
  + **Visszatérési érték:**
    - Egy JSON válasz, amely tartalmazza az összes rendelkezésre álló tápegységet.
  + **Leírás:** Ez a metódus visszaadja az összes rendelkezésre álló tápegységet.
* **show($id: int): JSON**
  + **Paraméter:**
    - **$id**: A lekérni kívánt tápegység azonosítója.
  + **Visszatérési érték:**
    - Egy JSON válasz a lekért tápegységgel.
  + **Leírás:** Ez a metódus visszaadja a megadott azonosítójú tápegységet.

A **PSUEfficiencyController** felelős a tápegység hatékonyság típusainak kezeléséért.

Metódusok:

* **index(): JSON**
  + **Visszatérési érték:**
    - Egy JSON válasz, amely tartalmazza az összes rendelkezésre álló tápegység hatékonyság típusát.
  + **Leírás:** Ez a metódus visszaadja az összes rendelkezésre álló tápegység hatékonyság típusát.
* **show($id: int): JSON**
  + **Paraméter:**
    - **$id**: A lekérni kívánt tápegység hatékonyság típusának azonosítója.
  + **Visszatérési érték:**
    - Egy JSON válasz a lekért tápegység hatékonyság típusával.
  + **Leírás:** Ez a metódus visszaadja a megadott azonosítójú tápegység hatékonyság típusát.

A **PSUModularController** felelős a tápegység modularitásának kezeléséért.

Metódusok:

* **index(): JSON**
  + **Visszatérési érték:**
    - Egy JSON válasz, amely tartalmazza az összes rendelkezésre álló tápegység modularitását.
  + **Leírás:** Ez a metódus visszaadja az összes rendelkezésre álló tápegység modularitását.
* **show($id: int): JSON**
  + **Paraméter:**
    - **$id**: A lekérni kívánt tápegység modularitásának azonosítója.
  + **Visszatérési érték:**
    - Egy JSON válasz a lekért tápegység modularitásával.
  + **Leírás:** Ez a metódus visszaadja a megadott azonosítójú tápegység modularitását.

A **PSUTypeController** felelős a tápegység típusának kezeléséért.

Metódusok:

* **index(): JSON**
  + **Visszatérési érték:**
    - Egy JSON válasz, amely tartalmazza az összes rendelkezésre álló tápegység típusát.
  + **Leírás:** Ez a metódus visszaadja az összes rendelkezésre álló tápegység típusát.
* **show($id: int): JSON**
  + **Paraméter:**
    - **$id**: A lekérni kívánt tápegység típusának azonosítója.
  + **Visszatérési érték:**
    - Egy JSON válasz a lekért tápegység típusával.
  + **Leírás:** Ez a metódus visszaadja a megadott azonosítójú tápegység típusát.

A **ResponseController** felelős a válaszok küldéséért az alkalmazás különböző részeiből.

Metódusok:

* **sendResponse($data: mixed, $message: string): JSON**
  + **Paraméterek:**
    - **$data**: Az adat, amelyet a válasz tartalmaz.
    - **$message**: Egy üzenet, amelyet a válaszban megjelenít.
  + **Visszatérési érték:**
    - Egy JSON válasz, amely tartalmazza a sikeres választ, az adatokat és az üzenetet.
  + **Leírás:** Ez a metódus elküldi a sikeres választ, amely tartalmazza az adatokat és az üzenetet.
* **sendError($error: string, $errorMessages: array, $code: int): JSON**
  + **Paraméterek:**
    - **$error**: Az hibaüzenet.
    - **$errorMessages**: További hibaüzenetek tömbje (opcionális).
    - **$code**: Az HTTP válasz kódja (alapértelmezett: 404).
  + **Visszatérési érték:**
    - Egy JSON válasz, amely tartalmazza a sikertelenséget, az hibaüzenetet és a hibakódot (opcionális).
  + **Leírás:** Ez a metódus elküldi a sikertelen választ, amely tartalmazza az hibaüzenetet és a hibakódot (ha van).

A **SidePanelTypesController** felelős a gépház oldalpanel típusainak kezeléséért.

Metódusok:

* **index(): JSON**
  + **Paraméterek:** Nincsenek
  + **Visszatérési érték:** Egy JSON válasz, amely tartalmazza az összes gépház oldalpanel típusát.
  + **Leírás:** Ez a metódus lekéri és visszaadja az összes gépház oldalpanel típusát.
* **show($id: int): JSON**
  + **Paraméterek:**
    - **$id**: A kívánt gépház oldalpanel típus azonosítója.
  + **Visszatérési érték:** Egy JSON válasz, amely tartalmazza a megadott azonosítójú gépház oldalpanel típust.
  + **Leírás:** Ez a metódus lekéri és visszaadja a megadott azonosítójú gépház oldalpanel típust.

A **UserController** felelős a felhasználók kezeléséért, beleértve a regisztrációt, bejelentkezést és kijelentkezést.

Metódusok:

* **register($request: UserRegisterChecker): JSON**
  + **Paraméterek:**
    - **$request**: Egy UserRegisterChecker típusú validált kérés objektum, amely tartalmazza a regisztrációhoz szükséges adatokat.
  + **Visszatérési érték:** Egy JSON válasz, amely jelzi a regisztráció sikerességét vagy hibáját.
  + **Leírás:** Ez a metódus regisztrálja az új felhasználót a kapott adatok alapján.
* **login($request: UserLoginChecker): JSON**
  + **Paraméterek:**
    - **$request**: Egy UserLoginChecker típusú validált kérés objektum, amely tartalmazza a bejelentkezéshez szükséges adatokat.
  + **Visszatérési érték:** Egy JSON válasz, amely jelzi a bejelentkezés sikerességét vagy hibáját.
  + **Leírás:** Ez a metódus kezeli a felhasználó bejelentkezési folyamatát, figyelembe véve a sikeres és sikertelen bejelentkezéseket.
* **logout($request: Request): JSON**
  + **Paraméterek:**
    - **$request**: Egy Request típusú objektum.
  + **Visszatérési érték:** Egy JSON válasz, amely jelzi a kijelentkezés sikerességét vagy hibáját.
  + **Leírás:** Ez a metódus kijelentkezteti a felhasználót és törli az aktuális hozzáférési tokenjét.

**Request-ek**

A **UserLoginChecker** osztály felelős a felhasználó bejelentkezési adatok validálásáért.

Metódusok:

* **authorize(): bool**
  + **Leírás:** Meghatározza, hogy a felhasználónak engedélyezett-e a kérés végrehajtása.
* **rules(): array**
  + **Visszatérési érték:** Egy asszociatív tömb, amely tartalmazza a validációs szabályokat.
  + **Leírás:** Visszaadja a bejelentkezési adatokra vonatkozó validációs szabályokat.
* **messages(): array**
  + **Visszatérési érték:** Egy asszociatív tömb, amely tartalmazza a validációs hibaüzeneteket.
  + **Leírás:** Visszaadja a validációs hibaüzeneteket a megfelelő nyelven.
* **failedValidation($validator: Validator)**
  + **Paraméterek:**
    - **$validator**: A Validator példány, amely tartalmazza a validálás eredményeit.
  + **Leírás:** Ez a metódus hívódik meg, ha a validáció sikertelen. Egy HTTP választ ad vissza a hibaüzenettel.

A **UserRegisterChecker** osztály felelős az új felhasználók regisztrációs adatok validálásáért.

Metódusok:

* **authorize(): bool**
  + **Leírás:** Meghatározza, hogy a felhasználónak engedélyezett-e a kérés végrehajtása.
* **rules(): array**
  + **Visszatérési érték:** Egy asszociatív tömb, amely tartalmazza a validációs szabályokat.
  + **Leírás:** Visszaadja a regisztrációs adatokra vonatkozó validációs szabályokat.
* **messages(): array**
  + **Visszatérési érték:** Egy asszociatív tömb, amely tartalmazza a validációs hibaüzeneteket.
  + **Leírás:** Visszaadja a validációs hibaüzeneteket a megfelelő nyelven.
* **failedValidation($validator: Validator)**
  + **Paraméterek:**
    - **$validator**: A Validator példány, amely tartalmazza a validálás eredményeit.
  + **Leírás:** Ez a metódus hívódik meg, ha a validáció sikertelen. Egy HTTP választ ad vissza a hibaüzenettel.

**Mail osztály(ok)**

Az **AlertMail** osztály felelős figyelmeztető e-mailek küldéséért. Ez **NEM** funkcionális a projektben.

Metódusok:

* **\_\_construct($content: string)**
  + **Paraméterek:**
    - **$content**: Az e-mail tartalma.
  + **Leírás:** Az osztály konstruktora, beállítja az e-mail tartalmát.
* **envelope(): Envelope**
  + **Visszatérési érték:** Az e-mail borítékolását leíró Envelope objektum.
  + **Leírás:** Visszaadja az e-mail borítékolását beállító Envelope objektumot.
* **content(): Content**
  + **Visszatérési érték:** Az e-mail tartalmát leíró Content objektum.
  + **Leírás:** Visszaadja az e-mail tartalmát beállító Content objektumot.
* **attachments(): array**
  + **Visszatérési érték:** Egy tömb, amely az e-mailhez csatolt fájlokat tartalmazza.
  + **Leírás:** Visszaadja az e-mailhez csatolt fájlokat.

**Modellek**

* CaseTypeModel
* Color
* ConfigModel
* CPUCoolerModel
* CPUModel
* GPUMemoryModel
* GPUModel
* IHDCapacityModel
* IHDFormFactorModel
* IHDInterfaceModel
* IHDModel
* IHDTypeModel
* MemoryModel
* MemoryModulesModel
* MOBOFFModel
* MOBOMMModel
* MOBOModel
* MOBOMSModel
* PC\_CaseModel
* PCPickerModelTest (üres teszt model, nem használt!)
* PSUEfficiencyModel
* PSUModularModel
* PSUTypeModel
* SidePanelTypesModel
* UserModel

**Seeder-ek**

Mindegyik Seeder (ami nem al-tábla [pl: Color, SidePanelTypes, GPUMemory]) maximum 50 adatot tároltat el, mindegyik adat .csv fájlokból van beolvasva. Ezeknek a forrása [itt található](https://github.com/docyx/pc-part-dataset).  
  
A CPUSeeder az egyetlen ami szimplább struktúrával rendelkezik mivel az az osztály egy saját kezűleg formázott .csv-t olvas be, míg a többi az “unprocessedpaths” mappában lévő fájlokból olvas be.  
  
Seeder lista (betöltési sorrendben):

* ColorSeeder
* CaseTypeSeeder
* SidePanelTypesSeeder
* PC\_CaseSeeder
* CPUSeeder
* CPU\_CoolerSeeder
* GPU\_MemorySeeder
* GPUSeeder
* IHD\_CapacitySeeder
* IHD\_FormFactorSeeder
* IHD\_InterfaceSeeder
* IHD\_TypeSeeder
* IHDSeeder
* MemoryModulesSeeder
* MemorySeeder
* MOBOFFSeeder
* MOBOMMSeeder
* MOBOMSSeeder
* MOBOSeeder
* PSUEfficiencySeeder
* PSUModularSeeder
* PSUTypeSeeder
* PSUSeeder

Ezt a listát sorrendbe meghívja a “DatabaseSeeder” osztály.

**API útvonalak**

Az **api.php** fájl tartalmazza az alkalmazás API végpontjainak útvonalbeállításait. Ez a dokumentum áttekintést nyújt a elérhető útvonalakról és azok megfelelő funkcionalitásairól.

**Hitelesítés**

Azon útvonalak, amelyek hitelesítést igényelnek, a Sanctum middleware-rel védettek.

**Hitelesített útvonalak**

* **/newconfig** (POST): Új konfiguráció létrehozása.
* **/modifyconfig** (PUT): Létező konfiguráció módosítása.
* **/deleteconfig** (DELETE): Konfiguráció törlése.
* **/logout** (POST): Az aktuális felhasználó kijelentkeztetése.
* **/configs** (GET): Az összes konfiguráció lekérése.
* **/configs/{id}** (GET): Egy adott konfiguráció lekérése ID alapján.

**Hitelesítés nélküli útvonalak**

* **/register** (POST): Új felhasználó regisztrálása.
* **/login** (POST): Bejelentkezés egy meglévő felhasználóval.

**Erőforrás útvonalak**

Az erőforrás útvonalak különböző entitásokhoz nyújtanak CRUD műveleteket az alkalmazásban.

**Builder komponenshez indexelt útvonalak**

* **/case\_type** (GET): Az összes ház típusának lekérése.
* **/colors** (GET): Az összes szín lekérése.
* **/cpu** (GET): Az összes CPU lekérése.
* **/cpu\_cooler** (GET): Az összes CPU hűtő lekérése.
* **/gpu\_memory** (GET): Az összes GPU memória lekérése.
* **/gpu** (GET): Az összes GPU lekérése.
* **/ihd\_capacity** (GET): Az összes belső merevlemez kapacitásának lekérése.
* **/ihd\_form\_factor** (GET): Az összes belső merevlemez formátumának lekérése.
* **/ihd\_interface** (GET): Az összes belső merevlemez interfész lekérése.
* **/ihd\_type** (GET): Az összes belső merevlemez típusának lekérése.
* **/internal\_hard\_drive** (GET): Az összes belső merevlemez lekérése.
* **/memory\_modules** (GET): Az összes memóriamodul lekérése.
* **/memory** (GET): Az összes memória lekérése.
* **/mobo\_form\_factor** (GET): Az összes alaplap formátumának lekérése.
* **/mobo\_max\_memory** (GET): Az összes alaplap maximális memóriakapacitásának lekérése.
* **/mobo\_memory\_slots** (GET): Az összes alaplap memóriahelyének lekérése.
* **/motherboard** (GET): Az összes alaplap lekérése.
* **/side\_panel\_types** (GET): Az összes oldalpanel típus lekérése.
* **/pc\_case** (GET): Az összes PC ház lekérése.
* **/psu\_efficiency** (GET): Az összes PSU hatékonyság lekérése.
* **/psu\_modular** (GET): Az összes PSU modularitás lekérése.
* **/psu\_type** (GET): Az összes PSU típus lekérése.
* **/psu** (GET): Az összes PSU lekérése.

Felhasználói dokumentáció

Ez a weboldal elsősorban a számítógép építés iránti rajongóknak lett létrehozva, de kezdőknek vagy a hobbi iránt érdeklődő felhasználóknak is kedvező lehet. Az oldal felépítése során törekedtünk a minél egyszerűbb, azaz “user friendly” felület megteremtésére.   
Korosztályt illetően bárki használhatja az oldalt, hiszen nagyon szimpla a használata.

A termék haszna hogy a felhaszálónak nincs szüksége a jegyzettömbre vagy egy noteszre, ahol felírja milyen alkatrészek alapján is állította össze a gépet, hanem egy felhasználó fiók használatával láthatja már az eddigi összeállított konfigurációkat, azokat tudja módosítani, illetve törölni.

Az oldal elindítása után a felhasználó a főoldallal fog elsősorban találkozni, ahol tud olvasni egy pár mondatot az oldal céljáról. Ezt követően az oldal felső sorjában találkozni fog pár lehetőséggel.  
A “How to” nevezetű oldalon tudunk olvasni egy számítógép összerakásáról, részreletesen leírva és tippeket adva az olvasás során, illetve képekkel illusztrálva minden egyes lépés.

A jobb felső sarokban a van egy “Sign Up” és egy “Log in” gomb, ahol betudunk lépni a már meglévő fiókunkba vagy akár létre is tudunk hozni egyet. A fiók létrehozásának annyi előnye van hogy az összerakott gépeket eltudjuk menteni és később tudjuk ezt módosítani vagy törölni és újat létrehozni. A fiók létrehozása után ajánlatos a számítógépünket összerakni a fentiek említettek miatt.

Következő oldal, ami az oldal szíve, az pedig a “Build it!” nevezetű oldal. Itt tudjuk létrehozni a konfigurációnkat és elmenteni a fiókunkra.   
A felhasználó nagyon sok alkatrész közül tud választani kategóriákra (Processzor, Alaplap, Videókártya…stb) felbontva, ezeket kiválasztva.  
A felhasználó akár több számítógép konfigurációt is tud elmenteni, ezeket a későbbiekben át tudja tekinteni a “My Configs” oldalon, amit szintén a oldal tetején talál meg.

Az oldal témáját követve nagyon lényegretörően lett létrehozva ez az oldal is. Kategóriákra szedve láthatjuk az elmentett alkatrészeket és minden egyes mentett konfiguráció végén megtalálható két gomb, a “Modify” gombbal tudjuk módosítani a kiválasztott konfigot. A gomb hatására visszairányít a “Build it!” oldalra, ahol tudjuk módosítani és újra elmenteni a kiválasztott gépet.   
A “Modify”gomb mellett található egy “Delete” gomb, amivel értelem szerűen törölni tudjuk az összerakott PCt-t.

Minden oldal alján megtalálható egy “footer”, ahol van egy rövid leírás az oldalról és kontakt információ (Github profil linkek, email címek és telefonszámok).

Frontend Fejlesztői dokumentáció

**UserConfigsComponent Dokumentáció**

Cél:

A **UserConfigsComponent** felelős a felhasználó által létrehozott konfigurációk megjelenítéséért. A felhasználó konfigurációit lekéri a háttérből a **ConfigService** segítségével, és táblázatos formában jeleníti meg őket, lehetővé téve a felhasználók számára, hogy módosítsák vagy töröljék a konfigurációkat.

Leírás:

* A **UserConfigsComponent** egy Angular komponens, amely megjeleníti a felhasználói konfigurációk listáját.
* A háttérből a **ConfigService** segítségével tölti be a felhasználói konfigurációkat.
* A betöltött konfigurációkat táblázatos formában jeleníti meg, minden konfigurációhoz olyan részleteket mutatva, mint a ház, CPU, CPU hűtő, GPU, belső merevlemez, memória, alaplap és tápegység.
* A komponens lehetőséget biztosít minden konfiguráció módosítására vagy törlésére gombok segítségével a táblázatban.

Sablon:

* Az HTML sablon (**userconfigs.component.html**) egy táblázatot tartalmaz, amely megjeleníti a felhasználói konfigurációkat.
* A táblázat egy sorban megjeleníti az egyes konfigurációkat, különböző komponensekhez tartozó oszlopokkal, valamint lehetőségeket biztosít a konfigurációk módosítására vagy törlésére.
* A táblázatot az Angular **\*ngFor** direktívával tölti ki, amely az konfigurációk listájának iterálására szolgál.

Komponens Osztály:

* A komponens osztálya (**UserConfigsComponent**) inicializálja a konfigurációk listáját, és betölti őket a háttérből az inicializálás során a **ConfigService** segítségével.
* A konfigurációk módosításának és törlésének kezelésére szolgáló módszereket biztosít, amelyeket a felhasználói műveletek indítanak el.
* A **loadConfigs** módszer lekéri a felhasználói konfigurációkat a háttérből, és frissíti a komponens adatait.
* A **modifyConfig** módszer a kiválasztott konfiguráció azonosítójával navigál a módosító oldalra.
* A **deleteConfig** módszer egy konfigurációt töröl az azonosítója alapján, és sikeres törlés után frissíti a konfigurációk listáját.

Használat:

* A **UserConfigsComponent** automatikusan betöltődik és megjelenik, amikor a felhasználó a felhasználói konfigurációk oldalra navigál.
* Felhasználóbarát felületet biztosít a felhasználó által létrehozott konfigurációk megtekintéséhez és kezeléséhez.
* A felhasználók közvetlenül ezen az oldalon módosíthatják vagy törölhetik a konfigurációkat, ami egy zökkenőmentes felhasználói élményt nyújt.

**HeaderComponent**

Cél:

A **HeaderComponent** felelős a PC-összeállító webhely fejléc szekciójának megjelenítéséért. Tartalmazza az oldal logóját, navigációs linkeket és az azonosításhoz kapcsolódó linkeket.

Leírás:

* A **HeaderComponent** két fő szakaszból áll: a bal szakaszból (**header-left**) és a jobb szakaszból (**header-right**).
* A bal szakaszban "Hogyan kell" és "Építsd meg!" navigációs linkek találhatók, amelyek a megfelelő oldalakra mutatnak.
* A jobb szakaszban az azonosításhoz kapcsolódó linkek vannak:
  + Ha a felhasználó nincs bejelentkezve, "Bejelentkezés" és "Regisztráció" linkek jelennek meg.
  + Ha a felhasználó be van jelentkezve, "Kijelentkezés" és "Saját Konfigurációim" linkek jelennek meg.
* A **isLoggedIn()** metódus ellenőrzi, hogy a felhasználó be van-e jelentkezve a **AuthService**-ból származó token meglétének ellenőrzésével.
* A **logout()** metódus kijelentkezteti a felhasználót a **AuthService** **logout()** metódusának meghívásával, majd átirányítja a felhasználót a főoldalra.

Használat:

A **HeaderComponent** használatához illessze be a **<app-header></app-header>** címkét bármelyik komponens sablonába, ahol meg akarja jeleníteni a fejlécet.

**HomeComponent**

Cél:

A **HomeComponent** felelős a PCBuilder webhely főoldalának megjelenítéséért. Ez a komponens bemutatja a webhely főbb jellemzőit és szolgáltatásait, valamint lehetőséget nyújt a PC építésének megkezdésére.

Leírás:

* A **HomeComponent** főleg két részből áll: a hős szekcióból (**hero**) és a jellemzők szekciójából (**features**).
* A hős szekcióban található egy cím, egy rövid leírás és egy "Építés megkezdése" gomb, amely a PC összeállító oldalra mutat.
* A jellemzők szekció három részből áll, mindegyik egy képet, egy címet és egy rövid leírást tartalmaz:
  + Széles körű alkatrészek: Válasszon a processzorok, alaplapok, grafikus kártyák és egyebek közül.
  + Egyszerű testreszabás: Testre szabhatja a PC-t az Ön konkrét igényeinek és költségvetésének megfelelően.
  + Szakértői támogatás: Segítséget kaphat egy szakértőkből álló csapatunktól, hogy tökéletes PC-t építsen.

Használat:

A **HomeComponent** automatikusan megjelenik, amikor a felhasználó az alkalmazást elindítja vagy a főoldalra navigál.

**BuilderComponent**

Cél:

A BuilderComponent felelős az alkatrészek kiválasztásáért és a konfiguráció összeállításáért a PC összeállító oldalon. Ez a komponens lehetővé teszi a felhasználók számára, hogy különböző alkatrészeket válasszanak, és menthessék a konfigurációjukat.

Leírás:

* A BuilderComponent egy konténerben (container) és egy fő részben (row) található.
* A fő részben található egy cím (Choose your parts), valamint egy lista (list-group) az alkatrészek kiválasztásához.
* A lista tartalmazza a következő alkatrészeket:
  + Processzor (Processor)
  + Processzor hűtő (Processor Cooler)
  + Memória (Memory)
  + Alaplap (Motherboard)
  + Grafikus kártya (Graphics Card)
  + Merevlemez / Tároló (Hard Drive / Storage)
  + Tápegység (Power Supply)
  + Ház (Case)
* Minden alkatrészhez egy legördülő lista (select) van hozzárendelve, amely a rendelkezésre álló lehetőségeket jeleníti meg.
* A felhasználó választhat az alkatrészek közül, majd elmentheti a konfigurációját, ha be van jelentkezve.  
    
    
    
  **FooterComponent**
* Cél:
* A **FooterComponent** felelős a PCBuilder webhely lábléceinek megjelenítéséért. Ez a komponens tartalmazza a webhely elérhetőségeit, hasznos linkeket, céginformációkat és a szerzői jogi információkat.
* Leírás:
* A **FooterComponent** egy láblécet tartalmaz két fő szakaszból áll:
* Az első szakaszban található egy szekció, amelyben elérhetők a közösségi média linkek.
* A második szakaszban található egy konténer, amely tartalmazza a vállalati információkat és hasznos linkeket.
* Elérhető közösségi média linkek:
* Facebook
* Twitter
* Google
* Instagram
* LinkedIn
* GitHub
* Vállalati információk:
* Cégnév: PCBuilder.com
* Cím: Budapest, Révay u. 16, 1065
* E-mail: jagerattila@gmail.com
* Telefonszám: + 06 20 642 5265
* Fax: + 01 70 135 7364
* Hasznos linkek:
* Miklós Rajmund GitHub: [Miklós Rajmund GitHub](https://github.com/ItsChippie)
* Rátkay Dániel GitHub: [Rátkay Dániel GitHub](https://github.com/Phreelosu)
* Segítség: [Segítség](https://chat.openai.com/)
* Szerzői jogi információ:
* A lábléc alján látható a szerzői jogi információ, amely jelzi, hogy a webhely tartalma 2024-ben a PCBuilder.com tulajdonát képezi.
* Használat:
* A **FooterComponent** automatikusan megjelenik az oldal alján, és a webhely minden oldalán jelen van.

**HowToComponent**

Cél:

A **HowToComponent** felelős az építési útmutató lépéseinek és tippeknek megjelenítéséért a PC összeállításról. Ez a komponens részletes lépéseket, tippeket és trükköket tartalmaz a PC összeállításához, valamint képeket a folyamatról.

Leírás:

* A **HowToComponent** egy konténerben található, amelyben számos részletes lépés és tipp található.
* A komponens képeket is tartalmaz, amelyek segítik az építési folyamat megértését.
* A felhasználók az oldal tetejére görgethetnek a "Back to Top" gombbal.

Elérhető képek:

1. PC összeállítás folyamata
2. Szükséges eszközök
3. Processzor
4. Processzor beszerelése
5. Processzor hűtő beszerelése
6. Folyadékhűtő rendszer
7. Hőpaszta alkalmazása
8. Processzor hűtő oldalnézete
9. Hűtő tartó beszerelése
10. Hűtő tartó alkatrészei
11. Hőpaszta alkalmazása más nézőpontból
12. CPU ventilátor csatlakozása
13. CPU ventilátor csatlakozása más nézőpontból
14. RAM beszerelése
15. RAM beszerelése más nézőpontból
16. M.2 SSD beszerelése
17. Alaplap kép
18. M.2 SSD beszerelése más nézőpontból
19. M.2 SSD beszerelése más nézőpontból
20. M.2 SSD csavar rögzítése
21. Alaplap építése
22. Alaplap távtartók
23. I/O pajzs
24. I/O pajzs más nézőpontból
25. Alaplap beszerelése a házba
26. Távtartók rögzítése
27. Kábelmenedzsment
28. Tápegység
29. Tároló
30. Grafikus kártya beszerelése
31. Grafikus kártya tartó
32. Grafikus kártya kábel
33. Grafikus kártya kábel csatlakoztatása
34. Grafikus kártya kábel más nézőpontból
35. Ház csatlakozók
36. Ház csatlakozók más nézőpontból
37. Be/Ki kapcsoló
38. Grafikus kártya beszerelése más nézőpontból
39. PC bekapcsolása
40. Kábelmenedzsment
41. Kábelmenedzsment más nézőpontból
42. Operációs rendszer telepítése
43. BIOS

Használat:

A **HowToComponent** automatikusan megjelenik, amikor a felhasználó a "How to" linkre kattint a fejlécben. A felhasználók lapozhatnak lefelé a teljes útmutató megtekintéséhez, és vissza is görgethetnek a lap tetejére a "Back to Top" gomb segítségével.

**LoginComponent**

Cél:

A **LoginComponent** felelős a felhasználó bejelentkezési folyamatának kezeléséért. Ez a komponens egy űrlapot biztosít a felhasználónév és jelszó megadásához, valamint lehetőséget nyújt a bejelentkezéshez.

Leírás:

* A **LoginComponent** egy űrlapot jelenít meg, amely tartalmazza a felhasználónév és jelszó mezőket, valamint egy "Remember me" jelölőnégyzetet.
* A felhasználók a megfelelő adatok megadása után be tudnak jelentkezni az űrlap elküldésével.
* Amennyiben a felhasználó sikeresen bejelentkezik, átirányításra kerülnek a "home" oldalra.

Elérhető funkciók:

1. Bejelentkezés: A felhasználó megadja az e-mail címét és a jelszavát, majd elküldi az űrlapot a bejelentkezéshez.
2. Megjegyzés: A felhasználó bejelölheti a "Remember me" jelölőnégyzetet, hogy az alkalmazás megjegyezze a bejelentkezési adatait.

Használat:

A **LoginComponent** automatikusan megjelenik, amikor a felhasználó a bejelentkezési oldalra navigál. A felhasználók kitöltik az űrlapot a megfelelő adatokkal, majd elküldik azt a bejelentkezéshez. Ha a bejelentkezés sikeres, az alkalmazás átirányítja a felhasználót a "home" oldalra.

**PublicComponent**

Cél:

A **PublicComponent** felelős a nyilvános oldalak tartalmának megjelenítéséért. Ez a komponens egy **<router-outlet>** elemet tartalmaz, amely lehetővé teszi a különböző nyilvános oldalak dinamikus megjelenítését.

Leírás:

* A **PublicComponent** egy üres komponens, amelynek célja a dinamikus tartalom megjelenítése az alkalmazás nyilvános oldalain.
* A **<router-outlet>** elem lehetővé teszi más komponensek (pl. **HomeComponent**, **LoginComponent**, stb.) dinamikus megjelenítését az alkalmazás nyilvános részein.

Elérhető funkciók:

1. Dinamikus tartalom megjelenítése: A **<router-outlet>** elem lehetővé teszi más komponensek dinamikus megjelenítését az alkalmazás nyilvános részein.

Használat:

A **PublicComponent** automatikusan használatra kerül, amikor a felhasználó a nyilvános oldalakat böngészi az alkalmazásban. A dinamikus tartalom megjelenítése érdekében más komponensek (pl. **HomeComponent**, **LoginComponent**, stb.) által meghatározott tartalom jelenik meg a **<router-outlet>** elemen keresztül.

**PublicModule**

Cél:

A **PublicModule** felelős a nyilvános oldalak komponenseinek betöltéséért és konfigurálásáért. Ez a modul tartalmazza a **PublicComponent**-et, amely a nyilvános oldalakat vezérli, valamint importálja és konfigurálja a szükséges egyéb modulokat és komponenseket.

Leírás:

* A **PublicModule** egy Angular modul, amely meghatározza a nyilvános oldalak komponenseit és konfigurációját.
* A **PublicComponent** a **PublicModule** része, és felelős a nyilvános oldalak tartalmának megjelenítéséért.
* A modul importálja és konfigurálja a **PublicRoutingModule**-et és a **LandingModule**-ot a megfelelő útválasztás és funkcionalitás biztosítása érdekében.

Elérhető funkciók:

1. Nyilvános oldalak komponenseinek betöltése: A modul lehetővé teszi a nyilvános oldalak komponenseinek betöltését és megjelenítését.
2. Útválasztás konfigurálása: A **PublicRoutingModule** importálása lehetővé teszi az útvonalak definiálását és kezelését a nyilvános oldalak között.
3. Egyéb modulok importálása és konfigurálása: A **LandingModule** importálása lehetővé teszi a kezdőlap moduljának használatát a nyilvános oldalakon.

Használat:

A **PublicModule** automatikusan betöltésre kerül az alkalmazás indításakor, és felelős a nyilvános oldalak komponenseinek konfigurálásáért és megjelenítéséért. A modul importálja a szükséges egyéb modulokat és komponenseket, és biztosítja azok használatát a nyilvános oldalak megjelenítéséhez.

**AppComponent**

Cél:

Az **AppComponent** az alkalmazás fő komponense, amely felelős az alkalmazás fő keretének létrehozásáért és megjelenítéséért. Ez az összetevő tartalmazza az alkalmazás fejlécét (**HeaderComponent**), láblécét (**FooterComponent**) és a fő tartalmi területet, ahol a különböző oldalak jelennek meg az alkalmazásban.

Leírás:

* Az **AppComponent** egy Angular komponens, amely az alkalmazás fő tartalmát kezeli és megjeleníti.
* Az alkalmazás fő tartalmi területeként szolgál, ahol az alkalmazás különböző részei megjelennek, például a kezdőlap, a felhasználói konfigurációk, a regisztrációs és bejelentkezési oldalak stb.
* A komponens tartalmazza az alkalmazás fejlécét (**HeaderComponent**) és láblécét (**FooterComponent**), amelyek az alkalmazás minden oldalán megjelennek.

Sablon:

* Az HTML sablon (**app.component.html**) tartalmazza az alkalmazás fő szerkezetét.
* Az **app-header** és **app-footer** komponensek beillesztése az alkalmazás fejlécét és láblécét biztosítja.
* A fő tartalomterületet a **router-outlet** komponens tölti be, amely az aktuális útválasztóútvonalnak megfelelő komponenst jeleníti meg.

Komponens Osztály:

* Az **AppComponent** osztály az alkalmazás fő komponense, amely vezérli az alkalmazás fő keretét.
* A komponens osztálya nem tartalmaz különösebb logikát, csak a címét (**title**) tartalmazza jelenleg.

Használat:

* Az **AppComponent** az alkalmazás fő kerete, amely minden oldalon megjelenik.
* Az alkalmazás minden részének megjelenítéséért és vezérléséért felelős.
* Az **AppComponent** tartalmazza az alkalmazás általános szerkezetét és stílusát, amelyen keresztül az alkalmazás felhasználói élménye koherens és összefüggő.

**AppModule**

Cél:

Az **AppModule** az Angular alkalmazás fő modulja, amely meghatározza az alkalmazás struktúráját és konfigurációját. Ez a modul tartalmazza az alkalmazás fő komponenseit, szolgáltatásait, és azokat a modulokat, amelyeket az alkalmazás használ.

Leírás:

* Az **AppModule** az Angular alkalmazás fő modulja, amely meghatározza az alkalmazás komponenseinek és moduljainak összetételét.
* A modul tartalmazza az alkalmazás fő komponenseit, mint például a kezdőlap (**HomeComponent**), a bejelentkezés (**LoginComponent**), az építő (**BuilderComponent**), a fejléc (**HeaderComponent**), a lábléc (**FooterComponent**) és egyebek.
* Az **AppModule** importálja azokat a modulokat, amelyeket az alkalmazás használ, mint például a **BrowserModule**, **HttpClientModule**, **AppRoutingModule**, **NgbModule** és mások.
* Az **AppModule** konfigurálja az alkalmazás gyökér komponensét (**AppComponent**) és szolgáltatásait.

Deklarációk:

* Az **AppModule** deklarációjában felsorolásra kerülnek az alkalmazás összes komponense (**declarations**), szolgáltatások, csővezetékek és egyéb elemek.
* Az itt felsorolt komponensek és egyéb elemek érhetőek el az alkalmazás különböző részein keresztül.

Modulok:

* Az **AppModule** importálja a következő modulokat: **BrowserModule**, **AppRoutingModule**, **NgbModule**, **ReactiveFormsModule**, **HttpClientModule**, **FormsModule**.
* A **BrowserModule** szükséges a böngésző specifikus funkciókhoz és az Angular alapvető szolgáltatásainak használatához.
* Az **AppRoutingModule** tartalmazza az alkalmazás útvonalainak definícióit.
* Az **NgbModule** a Bootstrap komponensek Angular specifikus implementációját biztosítja.
* A **ReactiveFormsModule** és a **FormsModule** szükségesek az űrlapokhoz és az adatok validálásához.

Szolgáltatók:

* Az **AppModule**-ban definiált szolgáltatók (**providers**) az alkalmazás számára globálisan elérhető szolgáltatásokat nyújtanak.
* Az itt felsorolt szolgáltatók érhetőek el az alkalmazás összes komponensében és szolgáltatásában.

Felhasználás:

* Az **AppModule** az Angular alkalmazás fő modulja, amely meghatározza az alkalmazás összetételét és konfigurációját.
* Ez az elsődleges modul, amelyet az Angular alkalmazás betölt, és ez tartalmazza az alkalmazás fő komponenseit és szolgáltatásait.
* Az **AppModule** felelős az alkalmazás felépítéséért és inicializálásáért, és meghatározza az alkalmazás struktúráját és működését.

**AuthService**

Cél:

Az **AuthService** szolgáltatás felelős a felhasználók be- és kijelentkezéséért, valamint az autentikációért és a token kezeléséért az Angular alkalmazásban.

Leírás:

* Az **AuthService** egy Angular szolgáltatás, amely felelős a felhasználók be- és kijelentkezéséért, valamint az autentikációért és a token kezeléséért az alkalmazásban.
* A szolgáltatás **HttpClient**-et használ a szerverrel történő kommunikációhoz, például a bejelentkezési kérelmek küldéséhez és a válaszok fogadásához.

Metódusok:

1. **login(username: string, password: string): Observable<any>**:
   * Felelősség: Ez a metódus elküldi a bejelentkezési adatokat a szervernek és visszatér a válasz Observable-jével.
   * Bemenetek:
     + **username**: A felhasználónév.
     + **password**: A jelszó.
   * Kimenet: Egy Observable, amely a szerver válaszát tartalmazza.
2. **logout(): void**:
   * Felelősség: Ez a metódus kijelentkezteti a felhasználót az alkalmazásból.
   * Működés: Törli a token-t a helyi tárolóból (**localStorage**).
3. **getToken(): string | null**:
   * Felelősség: Ez a metódus visszaadja a felhasználóhoz társított tokent az autentikációhoz.
   * Kimenet: A felhasználóhoz társított token, vagy null, ha nincs tárolva token.

Felhasználás:

* Az **AuthService** szolgáltatás segítségével lehet be- és kijelentkezni a felhasználókat az alkalmazásban.
* Az alkalmazás egyéb részei a **login()**, **logout()** és **getToken()** metódusokat használhatják az autentikációhoz és a token kezeléséhez.

**Config Model**

Cél:

A **Config** modell leírása, amely egy felhasználó konfigurációját reprezentálja az alkalmazásban.

Tulajdonságok:

1. **id: number**:
   * Leírás: A konfiguráció egyedi azonosítója.
   * Típus: Szám (number).
2. **case: number**:
   * Leírás: A konfigurációhoz tartozó ház azonosítója.
   * Típus: Szám (number).
3. **cpu: number**:
   * Leírás: A konfigurációhoz tartozó processzor azonosítója.
   * Típus: Szám (number).
4. **cpu\_cooler: number**:
   * Leírás: A konfigurációhoz tartozó processzor hűtő azonosítója.
   * Típus: Szám (number).
5. **gpu: number**:
   * Leírás: A konfigurációhoz tartozó grafikus kártya azonosítója.
   * Típus: Szám (number).
6. **internal\_hard\_drive: number**:
   * Leírás: A konfigurációhoz tartozó belső merevlemez azonosítója.
   * Típus: Szám (number).
7. **memory: number**:
   * Leírás: A konfigurációhoz tartozó memória modul azonosítója.
   * Típus: Szám (number).
8. **motherboard: number**:
   * Leírás: A konfigurációhoz tartozó alaplap azonosítója.
   * Típus: Szám (number).
9. **psu: number**:
   * Leírás: A konfigurációhoz tartozó tápegység azonosítója.
   * Típus: Szám (number).
10. **created\_at: string**:
    * Leírás: A konfiguráció létrehozásának dátuma és időpontja.
    * Típus: Szöveg (string), dátum formátumban.
11. **updated\_at: string**:
    * Leírás: A konfiguráció utolsó módosításának dátuma és időpontja.
    * Típus: Szöveg (string), dátum formátumban.
12. **user\_id: number**:
    * Leírás: A konfigurációhoz tartozó felhasználó azonosítója.
    * Típus: Szám (number).

Felhasználás:

* A **Config** modell segítségével lehet reprezentálni egy felhasználó által összeállított konfigurációt az alkalmazásban.
* Az alkalmazás egyéb részei, például a konfigurációt kezelő szolgáltatások és komponensek, használhatják ezt a modellt az adatok tárolására és megjelenítésére.

**UserConfigsComponent**

Cél:

A **UserConfigsComponent** felelős a felhasználó által létrehozott konfigurációk megjelenítéséért. A felhasználó konfigurációit lekéri a háttérből a **ConfigService** segítségével, és táblázatos formában jeleníti meg őket, lehetővé téve a felhasználók számára, hogy módosítsák vagy töröljék a konfigurációkat.

Leírás:

* A **UserConfigsComponent** egy Angular komponens, amely megjeleníti a felhasználói konfigurációk listáját.
* A háttérből a **ConfigService** segítségével tölti be a felhasználói konfigurációkat.
* A betöltött konfigurációkat táblázatos formában jeleníti meg, minden konfigurációhoz olyan részleteket mutatva, mint a ház, CPU, CPU hűtő, GPU, belső merevlemez, memória, alaplap és tápegység.
* A komponens lehetőséget biztosít minden konfiguráció módosítására vagy törlésére gombok segítségével a táblázatban.

Sablon:

* Az HTML sablon (**userconfigs.component.html**) egy táblázatot tartalmaz, amely megjeleníti a felhasználói konfigurációkat.
* A táblázat egy sorban megjeleníti az egyes konfigurációkat, különböző komponensekhez tartozó oszlopokkal, valamint lehetőségeket biztosít a konfigurációk módosítására vagy törlésére.
* A táblázatot az Angular **\*ngFor** direktívával tölti ki, amely az konfigurációk listájának iterálására szolgál.

Komponens Osztály:

* A komponens osztálya (**UserConfigsComponent**) inicializálja a konfigurációk listáját, és betölti őket a háttérből az inicializálás során a **ConfigService** segítségével.
* A konfigurációk módosításának és törlésének kezelésére szolgáló módszereket biztosít, amelyeket a felhasználói műveletek indítanak el.
* A **loadConfigs** módszer lekéri a felhasználói konfigurációkat a háttérből, és frissíti a komponens adatait.
* A **modifyConfig** módszer a kiválasztott konfiguráció azonosítójával navigál a módosító oldalra.
* A **deleteConfig** módszer egy konfigurációt töröl az azonosítója alapján, és sikeres törlés után frissíti a konfigurációk listáját.

Használat:

* A **UserConfigsComponent** automatikusan betöltődik és megjelenik, amikor a felhasználó a felhasználói konfigurációk oldalra navigál.
* Felhasználóbarát felületet biztosít a felhasználó által létrehozott konfigurációk megtekintéséhez és kezeléséhez.
* A felhasználók közvetlenül ezen az oldalon módosíthatják vagy törölhetik a konfigurációkat, ami egy zökkenőmentes felhasználói élményt nyújt.

**Config Service**

Cél:

A **ConfigService** osztály leírása, amely az alkalmazásban a felhasználó konfigurációinak kezelését végzi.

Metódusok:

1. **getUserConfigs(): Observable<Config[]>**:
   * Leírás: A felhasználó konfigurációinak lekérdezése a szerverről.
   * Visszatérési érték: Egy **Observable** objektum, amely egy **Config** tömböt tartalmaz.
2. **deleteConfig(configId: number): Observable<void>**:
   * Leírás: Egy konfiguráció törlése a szerverről.
   * Bemeneti paraméter: **configId** - A törlendő konfiguráció azonosítója.
   * Visszatérési érték: Egy **Observable** objektum, amely **void** típust tartalmaz.

Felhasználás:

* A **ConfigService** segítségével lehet kommunikálni a szerverrel, hogy lekérdezze vagy törölje a felhasználó konfigurációit.
* A komponensek és szolgáltatások ezt a szolgáltatást használhatják a konfigurációk kezelésére és megjelenítésére az alkalmazásban.

Fontos megjegyzés:

* A szolgáltatás az autentikációhoz szükséges token-t localStorage-ból olvassa ki, és hozzáadja a HTTP kérések fejlécéhez. Ez biztosítja, hogy csak bejelentkezett felhasználók férjenek hozzá a konfigurációkhoz.

**SignUpService**

Cél:

A **SignupService** osztály leírása, amely az alkalmazásban a regisztrációval és bejelentkezéssel kapcsolatos funkciókat nyújtja, valamint a szükséges alkatrészek lekérdezését a konfigurációkhoz.

Metódusok:

1. **signUp(data: any)**:
   * Leírás: Felhasználó regisztrációja a szerveren.
   * Bemeneti paraméter: **data** - Az új felhasználó adatait tartalmazó objektum.
   * Visszatérési érték: Egy **Observable** objektum, amely általában egy választ ad vissza a szervertől a sikeres regisztráció esetén.
2. **logIn(data: any)**:
   * Leírás: Felhasználó bejelentkezése a szerveren.
   * Bemeneti paraméter: **data** - A bejelentkezéshez szükséges adatokat tartalmazó objektum.
   * Visszatérési érték: Egy **Observable** objektum, amely általában egy választ ad vissza a szervertől a sikeres bejelentkezés esetén.
3. **getCPUs()**:
   * Leírás: Processzorok lekérdezése a szerverről.
   * Visszatérési érték: Egy **Observable** objektum, amely egy processzorokat tartalmazó tömböt ad vissza.
4. **getMotherboards()**:
   * Leírás: Alaplapok lekérdezése a szerverről.
   * Visszatérési érték: Egy **Observable** objektum, amely egy alaplapokat tartalmazó tömböt ad vissza.
5. **getMemory()**:
   * Leírás: Memóriák lekérdezése a szerverről.
   * Visszatérési érték: Egy **Observable** objektum, amely egy memóriákat tartalmazó tömböt ad vissza.
6. **getCoolers()**:
   * Leírás: Processzor hűtők lekérdezése a szerverről.
   * Visszatérési érték: Egy **Observable** objektum, amely egy processzor hűtőket tartalmazó tömböt ad vissza.
7. **getPSUs()**:
   * Leírás: Tápegységek lekérdezése a szerverről.
   * Visszatérési érték: Egy **Observable** objektum, amely egy tápegységeket tartalmazó tömböt ad vissza.
8. **getGPUs()**:
   * Leírás: Grafikus kártyák lekérdezése a szerverről.
   * Visszatérési érték: Egy **Observable** objektum, amely egy grafikus kártyákat tartalmazó tömböt ad vissza.
9. **getIHDs()**:
   * Leírás: Belső merevlemezek lekérdezése a szerverről.
   * Visszatérési érték: Egy **Observable** objektum, amely egy belső merevlemezeket tartalmazó tömböt ad vissza.
10. **getCases()**:

* Leírás: Számítógépházak lekérdezése a szerverről.
* Visszatérési érték: Egy **Observable** objektum, amely egy számítógépházakat tartalmazó tömböt ad vissza.

1. **saveConfiguration(configData: any)**:
   * Leírás: Felhasználó konfigurációjának mentése a szerveren.
   * Bemeneti paraméter: **configData** - A menteni kívánt konfigurációt leíró objektum.
   * Visszatérési érték: Egy **Observable** objektum, amely általában egy választ ad vissza a szervertől a sikeres mentés esetén.
2. **getSavedConfigurations()**:
   * Leírás: Elmentett konfigurációk lekérdezése a szerverről.
   * Visszatérési érték: Egy **Observable** objektum, amely egy elmentett konfigurációkat tartalmazó tömböt ad vissza.

Felhasználás:

* A **SignupService** osztály segítségével lehet regisztrálni, bejelentkezni, valamint lekérdezni a rendelkezésre álló alkatrészeket a konfigurációkhoz.
* A komponensek és szolgáltatások ezt a szolgáltatást használhatják a felhasználói műveletek végrehajtásához és az alkatrészek lekérdezéséhez.