Akasztófa

Fejlesztői dokumentáció

2021

Minta feladat

# Cél

Egy közösségi akasztófa játék készítése melynek lényege, hogy a játékos minél több pontot érjen el.

A backend-t Azure keretrendszer biztosítja. Frontendnek egy asztali, és egy webes alkalmazás áll rendelkezésre.

A webes alkalmazásban követhetik nyomon a játékosos az álltaluk elért pontszámot valamint regisztrált játékosok, képesek a szótárt bővíteni. Az asztali alkalmazásból bejelentkezést követően léphetnek be a játékosok és kezdhetnek neki a pontgyűjtésnek.

Felhasznált technológiák

Backend

Microsoft Azure

Készítéshez használt programok

dia

Visual studio 2019

Microsoft Azure

# Kódolási konvenciók

A kódot git verziókezelővel használjuk.

## Alapkönyvtárak

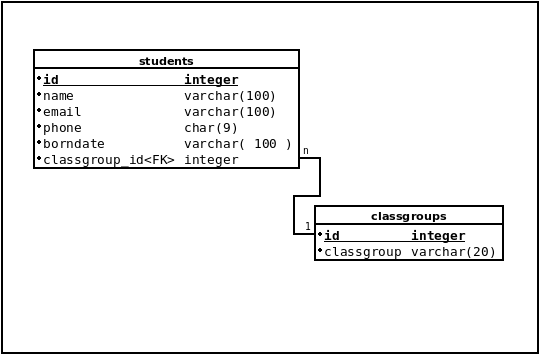
1. Webapp
2. Játék/Akasztofa
3. Docs

A webapp a web applikáció és backend a játék az asztali alkalmazás.

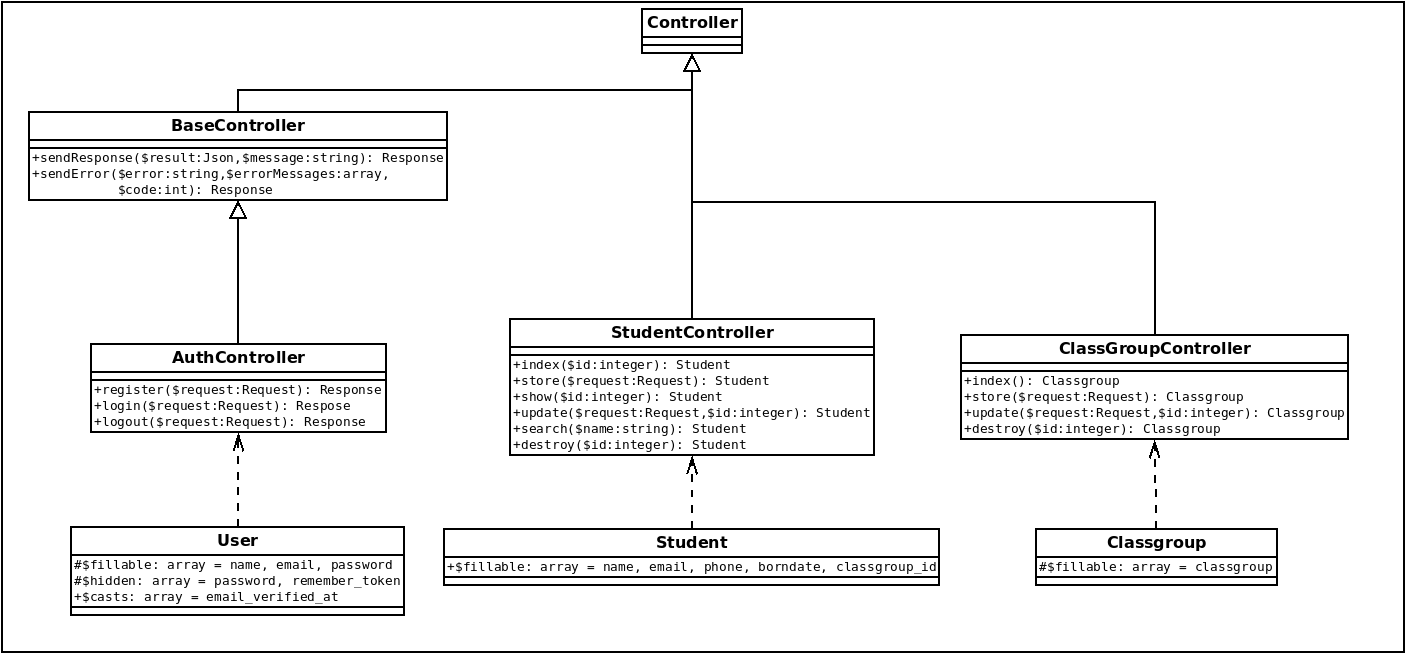
# 

# Felülettervek

## Adatmodell



API UML terv



Általános működés:

A REST API http kéréseket fogad, melyek tartalmazzák a műveletekhez szükséges megfelelő adatokat. A kényes műveletek végpontjai védettek, autentikáció szükséges a használatukhoz. Ilyenek a kijelentkezés, az új diák felvétele, diák adatainak módosítása, diák törlése, új csoport felvétele, csoport módosítása, csoport törlése. A regisztráció, a bejelentkezés, a diákok és csoportok lekérdezésének végpontjai publikusak, ezek használatához autentikáció nem szükséges.

Az adatokat Json formátumban fogadja és feldolgozza. A vezérlést kontrollerek valósítják meg, minden adatkezelési csoportnak ( autentikáció, diák, csoport ) külön kontrollere van, itt történik az adatfeldolgozás.

A kontrollerek modellekkel vannak kapcsolatban amelyek az adatkezelésért felelősek. Minden adatkezelési csoportnak ( felhasználó, diák, csoport ) külön modellje van, itt történik az adatok adatbázisból kiolvasása, illetve a az adatok kiírása adatbázisba.

A modellek adatbázis táblákkal vannak kapcsolatban, melyek az adatok tárolásáért felelősek. Az adatbázis táblák adatait a modellek kezelik.

## Osztályok

**BaseController**

Feladata a bejelentkezési műveletek válaszainak küldése. Sikeres művelet esetén a generált azonosító tokent visszaküldi, sikertelen művelet esetén hibaüzenetet küld.

Metódusok:

sendResponse()

- bejövő paraméterek: $result ( A generált token ), $message ( A felhasználónak szánt saját üzenet )

- kimenő adatok: Response ( sikeres bejelentkezés esetén a token ).

sendError()

-bejövő paraméterek: $error ( A php által generált hibaüzenet ), $errorMessages ( Saját hibaüzenet ), $code ( A válaszban küldendő http kód ).

-kimenő adatok: Response ( sikertelen bejelentkezés esetén hibaüzenet és a hiba kódja ).

**AuthController**

Feladata egy új felhasználó felvétele, felhasználók autentikációja, felhasználók kijelentkeztetése..

Metódusok:

register()

- bejövő paraméterek: $request ( az regisztrációhoz szükséges adatok a kérésben, name, email, password, confirm\_password )

Az adatokat validálja, majd sikeres érvényesítés után bejegyzi az users adatbázis tábla megfelelő mezőibe.

- kimenő adatok: saját üzenet.

login()

Feladata a felhasználó azonosítása név és jelszó alapján. Sikeres autentikáció esetén generál egy tokent a felhasználó számára és bejegyzi a personal\_access\_tokens adatbázis tábla megfelelő mezőjébe, majd átadja a BaseController sendResponse() metódusának a saját üzenettel együtt.

- bejövő paraméterek: $request ( a bejelentkezéshez szükséges adatok a kérésben, name, password ).

- kimenő adatok: token, saját üzenet.

logout()

Feladata a felhasználó kijelentkeztetése és a token törlése az adatbázis táblából.

- bejövő paraméterek: $request ( a kijelentkezéshez szükségek adatok a kérésben, token )

- kimenő adatok: saját üzenet.

**StudentController**

Feladata egy új diák felvétele, meglévő diák adatainak módosítása, meglévő diák keresése, meglévő diák törlése.

Metódusok:

index()

Visszaadja az adatbázisban szereplő diákok listáját csoportonként.

- bejövő paraméterek: NINCS

- kimenő adatai: Student ( diák modell amely tartalmazza a diákok adatbázisból lekért adatait.

store()

Eltárol az adatbázisban egy új diákot.

- bejövő paraméterek: request ( A diák adatait tartalmazó kérés ).

- kimenő adatok: Student ( A diák adatait tartalmazó modell ).

show()

Visszaadja egy kiválasztott diák adatait.

- bejövő paraméterek: $id ( a kiválasztott diák azonosítója )

- kimenő adatok: Student ( a kiválasztott diák adatait tartalmazó modell ).

update()

Frissíti az adatbázisban tárolt egy diák adatait.

- bejövő paraméterek: $request, $id ( a diák adatait tartalmazó kérés és a diák azonosítója ).

- kimenő adatok: Student ( a diák adatait tartalmazó modell ).

search()

Név vagy név részlet alapján kikeres egy vagy több diákot az adatbázisból.

- bejövő paraméterek: $name ( a diák neve vagy nevének részlete ).

- kimenő adatok: Student ( az egyezést mutató diákok adatait tartalmazó modell ).

destroy()

Feladata a kiválasztott diák törlése az adatbázisból.

- bejövő paraméterek: $id ( a kiválasztott diák azonosítója ).

- kimenő adatok: Student ( a kiválasztott diák adatait tartalmazó modell ).

**ClassGroupController**

Feladata a csoportok kezelése, új csoport felvétele, csoport módosítása, csoport törlése.

Metódusok:

index()

Visszaadja az adatbázisban szereplő csoportok listáját.

- bejövő paraméterek: NINCS

- kimenő adatai: Classgroup ( csoport modell amely tartalmazza a csoportok adatbázisból lekért adatait.

store()

Eltárol az adatbázisban egy új csoportot.

- bejövő paraméterek: request ( A csoport nevét tartalmazó kérés ).

- kimenő adatok: Student ( A csoport nevét tartalmazó modell ).

update()

Frissíti az adatbázisban tárolt egy csoport nevét.

- bejövő paraméterek: $request, $id ( a csoport nevét tartalmazó kérés és a csoport azonosítója ).

- kimenő adatok: ClassGroup ( a csoport nevét tartalmazó modell ).

destroy()

Feladata a kiválasztott csoport törlése az adatbázisból.

- bejövő paraméterek: $id ( a kiválasztott csoport azonosítója ).

- kimenő adatok: ClassGroup ( a kiválasztott csoport nevét tartalmazó modell ).

**User**

A regisztrált felhasználók adatait illetve az azonosító tokent amelynek a generálása a felhasználóhoz kapcsolódik írja az adatbázis megfelelő táblájába, valamint onnan szolgáltatja szükség szerint. A User modell a users adatbázis táblával van kapcsolatban az adatokat onnan olvassa és oda írja. Az osztályt a Laravel projekt generálja, használata a Sanctum autentikációs kiegészítő csomagon keresztül történik.

**Student**

Az adatbázisban felvett tanulók adatait kezeli, a StudentController megfelelő metódusai által kért adatot adja vissza az adatbázis megfelelő táblájából , illetve a kontrollerből érkező adatokat írja ki. Az adatbázis students táblájával van kapcsolatban, az adatokat onnan olvassa és oda írja.

Mezők:

$fillable ( array ) elemei: name, email, phone, borndate, classgrop\_id

A fillable mező elemei egy – egy mező a students adatbázis táblában. Biztonsági okokból csak a felsorolt mezőket írhatja a modell az adatbázisban.

**Classgroup**

Az adatbázisban felvett csoportok adatait kezeli, a ClassGroupController megfelelő metódusai által kért adatot adja vissza az adatbázis megfelelő táblájából , illetve a kontrollerből érkező adatokat írja ki. Az adatbázis classgroups táblájával van kapcsolatban, az adatokat onnan olvassa és oda írja.

Mezők:

$fillable ( array ) elemei: classgroup

A fillable mező eleme egy mező a classgroups adatbázis táblában. Biztonsági okokból csak a felsorolt mezőket írhatja a modell az adatbázisban.

# Fejlesztői környezet

* XAMPP
  + Adatbázis kezelése a mysql nevű klienssel.
  + PHP
* Visual Studio Code
* React Native Expo
* Laravel
* Angular
* Java FX

# Tesztelés

# Ismert hibák

Fejlesztési lehetőség