**Feladat:**

Készítsünk programot, amellyel az alábbi motoros játékot játszhatjuk. A feladatunk, hogy egy gyorsuló motorral minél tovább tudjunk haladni. A gyorsuláshoz a motor üzemanyagot fogyaszt, egyre többet. Adott egy kezdeti mennyiség, amelyet a játék során üzemanyagcellák felvételével tudunk növelni. A motorral a képernyő alsó sorában tudunk balra, illetve jobbra navigálni. A képernyő felső sorában meghatározott időközönként véletlenszerű pozícióban jelennek meg üzemanyagcellák, amelyek folyamatosan közelednek a képernyő alja felé. Mivel a motor gyorsul, ezért a cellák egyre gyorsabban fognak közeledni, és mivel a motor oldalazó sebessége nem változik, idővel egyre nehezebb lesz felvenni őket, így egyszer biztosan kifogyunk üzemanyagból. A játék célja az, hogy a kifogyás minél később következzen be. A program biztosítson lehetőséget új játék kezdésére, valamint játék szüneteltetésére (ekkor nem telik az idő, és nem mozog semmi a játékban). Ismerje fel, ha vége a játéknak, és jelenítse meg, mennyi volt a játékidő. Ezen felül szüneteltetés alatt legyen lehetőség a játék elmentésére, valamint betöltésére.

**Elemzés:**

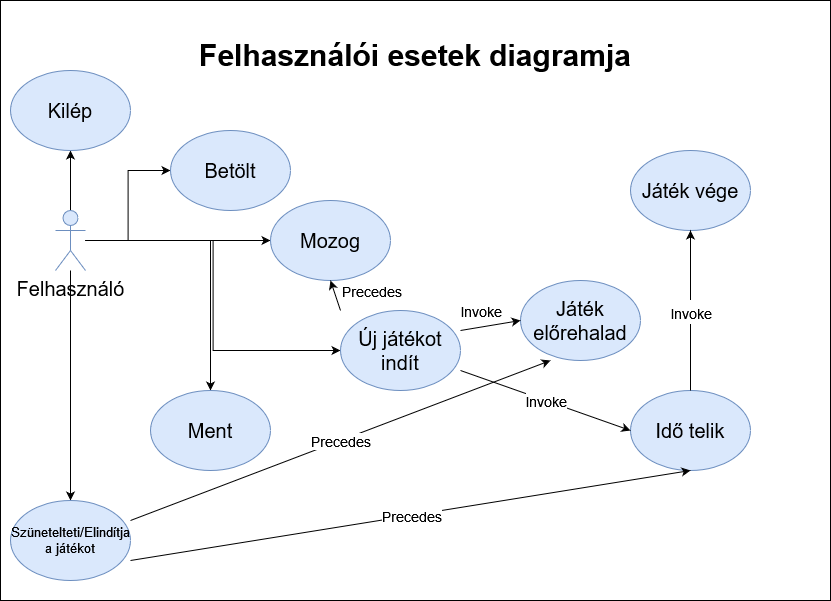
A játékban nem lehet nehézségi szintet állítani, de az idő múlásával gyorsulnia kell a motornak ami az üzemanyagcellák gyorsulását idézi elő, így nehezítve a játékmenetet.

A játéktáblát egy 9 × 5 mátrix reprezentálja, ami nyomógombokból áll, a legfelső sora kivételével. A balra illetve jobbra nyilak megnyomásával mozgathatjuk a motort a rács teljes szélességében a legalsó sorban. A tábla tetejéről hullanak le az üzemanyagcellák, amik eltűnnek ha felvesszük őket illetve ha „leesnek” a pályáról.

A játék egy 1 ablakos viszont 3 képernyős .NET MAUI grafikus felületű alkalmazás elsősorban Andrioid illetve Windows platformokra. Az első(/kezdő) képernyőn található maga a játék, a további képernyőkre pedig a Save illetve Load game gombok segítségével navigálhatunk. Ezeken elmenthejtük illetve betölthetjük az adott játékállásokat. Az kezdőképernyőn az ablak alján található két cimke, amik mutatják az eltelt időt, illetve hogy mennyi üzemanyagunk van.

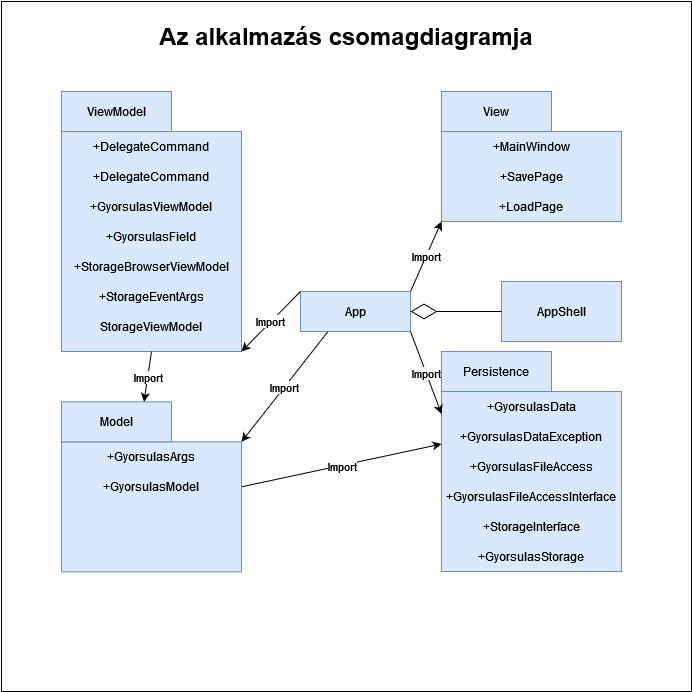
A játéktábla felett találhatóak az új játékért, a játék szüneteltetéséért, mentéséért illetve betöltéséért felelős gombok. Ezek alatt taláhatók a játékos mozgatásáért felelős gombok.

A játék automatikusan észleli hogy vége van-e a játéknak és előugró üzenettel jelzi amikor ez bekövetkezik. A mentést és betöltést a nekik megfelelő képernyőn kezeljük. Mentésnél beírhatunk egy saját fájlnevet amin mentünk, betöltésnél pedig ezen nevekre nyomva tudjuk előhozni a már megkezdett játékainkat. Ezen tevékenységek sikerességét szintén előugró üzenettel jelezzük.



**Tervezés:**

**Programszerkezet:**

A program az MVVM (Model,View, ViewModel, valamit a Persistence névtereket valósítjuk meg) architektúra alapján készült el. Ezen rétegek két külön projektbe vannak szervezve, a Model és valamint a View és ViewModel kerültek azonos projektekbe illetve mindkét projektben található 1-1 Persistence névtér. A program vezérlését az App osztály valósítja meg amely példányosítja és összeköti a modellt, nézetmodellt illetve a nézetet, valamint biztosítja a kommunkiációt.

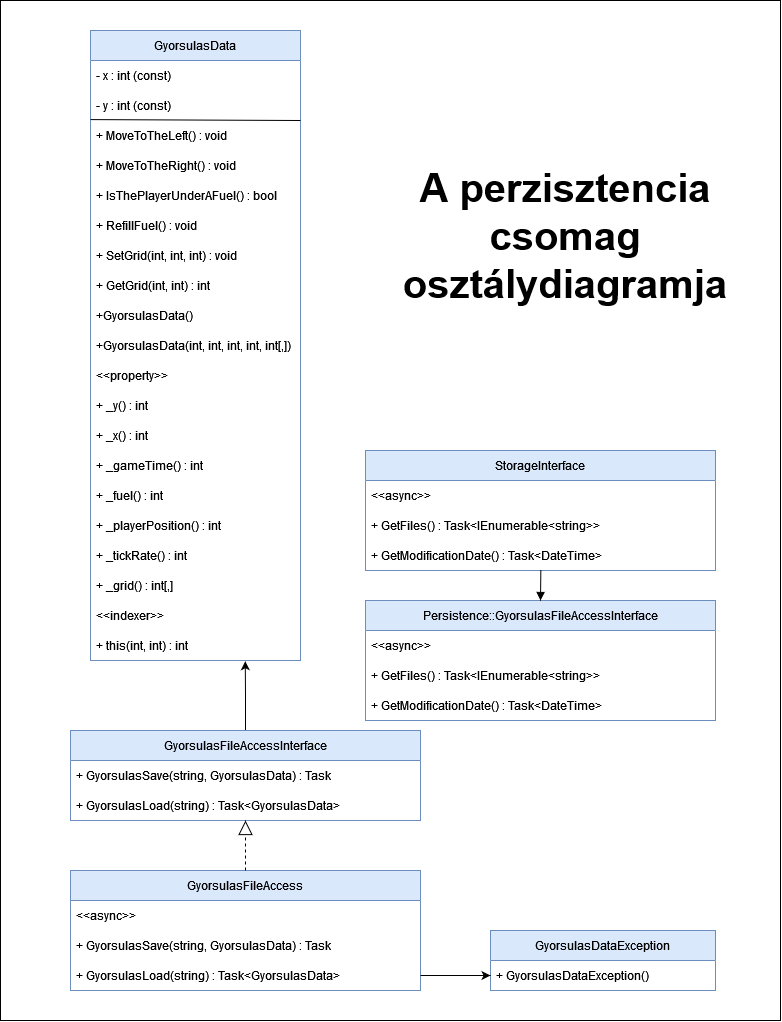
**Perzisztencia:**

Itt találatók a hosszútávú tárolásra szánt adatok, illetve a fájlkezelés lebonyolítása.

A ***GyorsulasData*** osztály tárolja a játékmezőt (\_grid) valamint ezen felül az összes olyan adatot (\_x, \_y, \_gameTime, \_fuel, \_playerPosition, \_tickRate) amire szükségünk van a játék működtetéséhez és a mentéshez illetve betöltéshez. Ezen adatok midnig a játék kezdeténél, új játéknál, betöltésnél generálódnak. Mindezen felül még lehetőséget biztosít a játékos feletti mező lekérdezésére, hogy tartalmaz-e üzemanyagot (IsThePlayerUnderAFuel) illetve a játékos mozgatására (MoveToTheLeft, MoveToTheRight), üzemanyag újratöltésére (RefillFuel), a pálya mezőinek módosítására (GetGrid, SetGrid).

A fájlkezelés a ***GyorsulasFileAccessInterface*** osztályon keresztül elérhetők. Ez a mentést (GyorsulasSave) és a betöltést (GyorsulasLoad) foglalja magában. Ezeket aszinkron módon bonyolítjuk le, hogy ne alakulhasson ki holtpont.

Az interfészt a ***GyorsulasFileAccess*** osztály implementálja, ami szöveges fájlok kezelésére szolgál. Ez az osztály hiba esetén ***GyorsulasDataException*** típusú hibával tér vissza. Sikeres mentéskor \*.gy kiterjesztésű fájlt generál. Ebben a fájlban az első sorban találhatók a tábla méretei, az eltelt idő, az üzemanyag mennyisége, a játékos pozíciója és a játék sebessége. A többi sor a játékmezőt reprezentálja, a 0 az üres mezőt, 1 a játékost és 2 az üzemanyagot.



**Modell:**

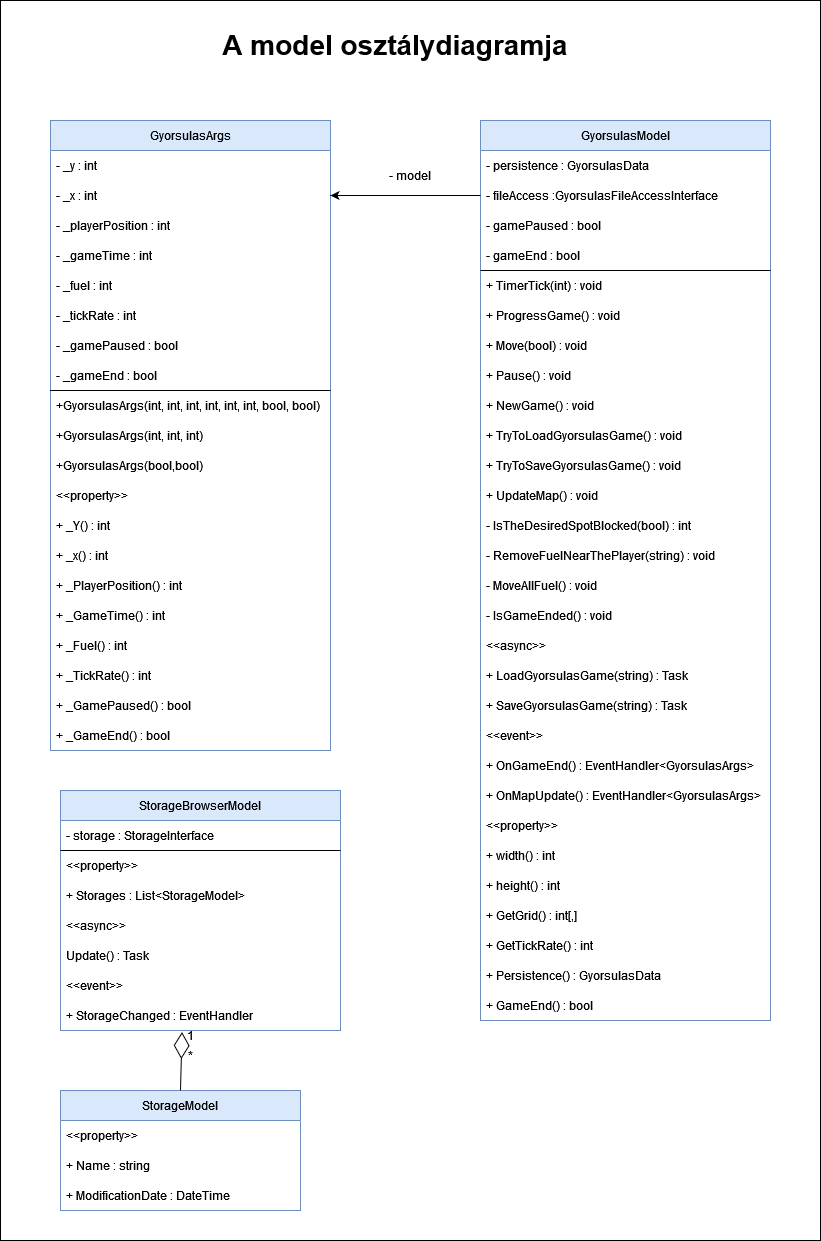
A modellt a ***GyorsulasModel*** és a ***GyorsulasArgs*** argumentumokat tároló osztályok valósítják meg. Az előbb említett osztály tartalmaz miden függvényt ami a játéktér állapotváltozásait képes szabályozni, illetve két változót (gameEnd, gamePaused) amik a tárolják hogy vége van-e a játéknak vagy az meg van-e éppen állítva.

A játékos mozgatásához (Move), a játék és az idő előreléptetéséhez (TimerTick, ProgressGame), a játéktábla frissítéséhez (UpdateMap) és új játék kezdéséhez (NewGame) tartozó események kezelése illetve maguknak a metódusoknak az implementálása zajlik itt.

A ***GyorsulasModel*** példányosításakor megkapja a ***GyorsulasFileAccessInterface*** felületet és ezen keresztül biztosítja a mentést (SaveGyorsulasGame) illetve a betöltést (LoadGyorsulasGame).

Az ***OnGameEnd*** esemény a játék végéről, az ***OnMapUpdate*** esemény pedig a játéktábla információinak frissitésének kérelméről tájékoztat.

A ***GyorsulasArgs*** tárolja az összes eddig felsorolt változót a perzisztenciából illetve a modellből, a játéktábla mátrixának kivételével.

****

**Nézetmodell:**

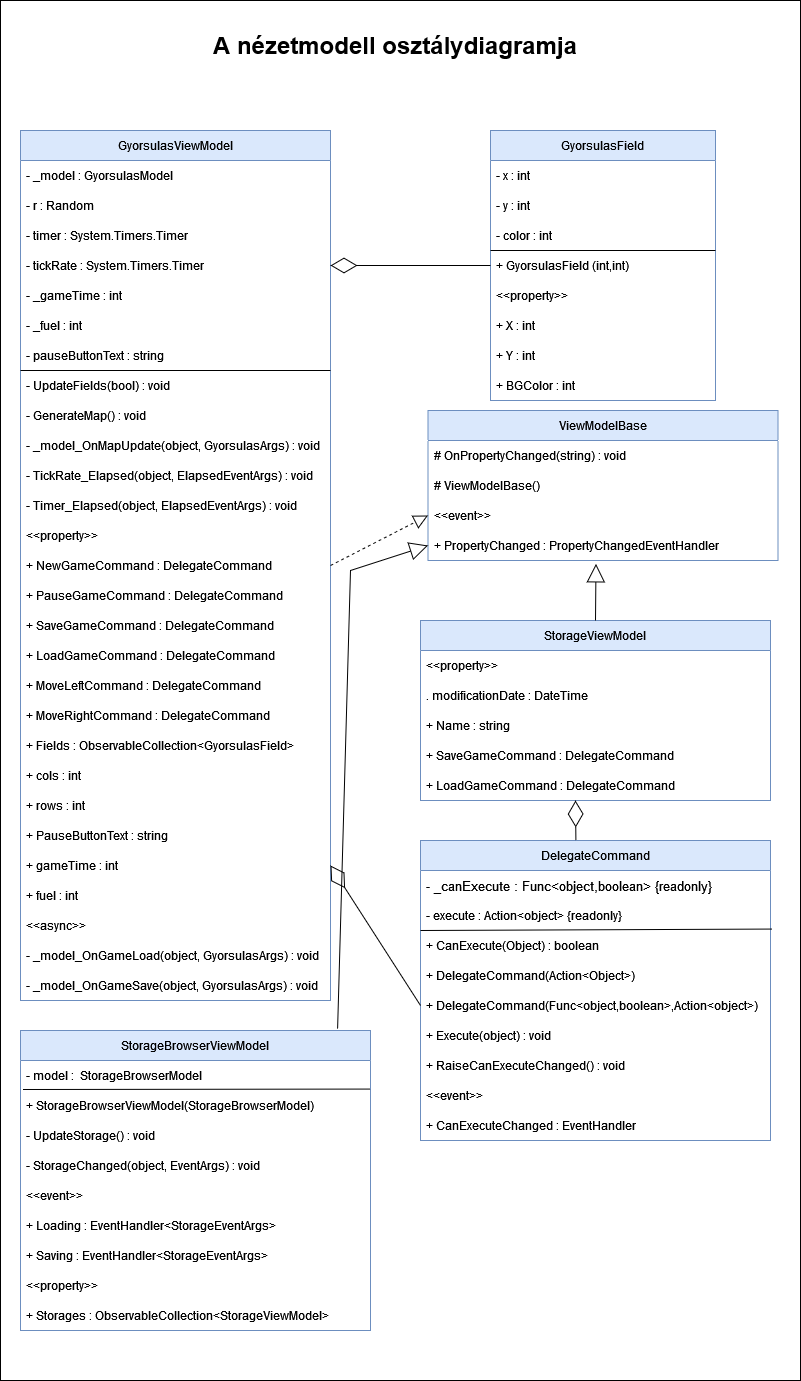
A nézetmodell megvalósításához felhasználunk egy általános utasítás (***DelegateCommand***), valamint egy ős változásjelző (***ViewModelBase***) osztályt.

A nézetmodell feladatait a GyorsulasViewModel látja el. Parancsokat biztosít a játék újrakezdésére, mentésére, betöltésére, szüneteltetésére illetve újrakezdésére. Itt van lekezelve a balra illetve jobbra nyilak lenyomása, ami kiváltja a játékos mozgatását.

A nézetmodell tárolja a modell egy hivatkozását információszerzés céljából, illetve tárolja még az adatkötéshez használt adatokat is. Tartalmaz még 2 timert (***timer***, ***tickRate***) és 1 random szám generátort (***r***), illetve az ezekhez tartozó eseménykezelőket, és a nézet frissítését a kötött adatok módosításával.

A játékmezők egy külön osztályban tároljuk. Itt az adott mező koordinátáit valamint a színét reprezentáló számot (***BGColor***) tároljuk el. Ezen mezőket egy Observable Collection szerkezetben tároljuk (***Fields***).

A tárolt játékállaptok mindegyikét egy StorageViewModel osztállyal írhatjuk le. Ezek kollekcióját dinamikusan, a fő nézetmodelltől függetlenül hozzuk létre, majd adjuk át a nézetnek.

****

**Nézet:**

A nézetet navigációs lapokkal valósítjuk meg:

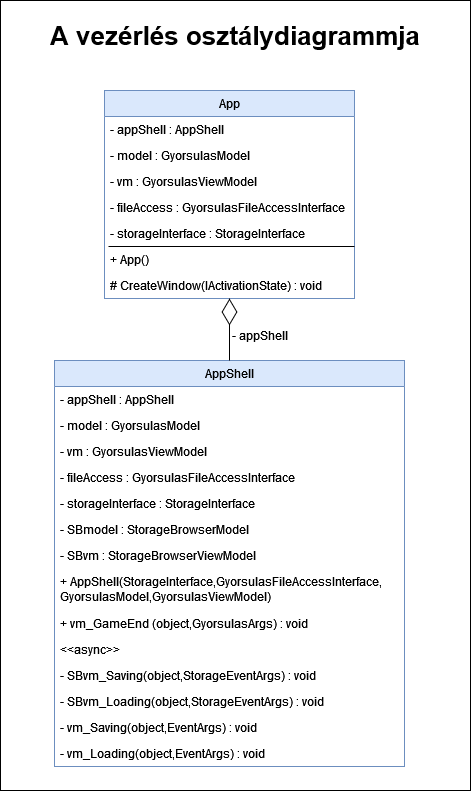
A ***MainPage*** tartalmazza a játéktáblát amely egy rácsból áll, amibe gombokat helyezünk. Ezen felül még 6 gombot tartalmaz amik rendre az új játékért, szüneteltetésért, mentésért betöltésért, balra illetve jobbra mozgásért felelősek.

A ***SavePage*** tartalmaz egy beviteli mezőt illetve egy gombot, amik a mentésért és a mentett fájl nevéért felelősek.

A ***LoadPage*** tartalmazza a már mentett játékok listáját amiket megérintve ezen játkállások betöltenek.

**Vezérlés:**

Az ***App*** osztály feladata a rétegek példányosítása valamint az alkalmazás vezérlése.

****Az alkalmazást az ***AppShell*** keretben helyezzük el, amely felelős a lapok közötti navigáció lebonyolításáért.

**Tesztelés:**

A ***GyorsulasTest*** osztály egységtesztek segítségével ellenőrzi a modell helyes működését az alábbi tesztekkel (minden fájlból való betöltés mockolt perzisztencia rétggel történik):

GyorsulasNewGameTest: ellenőrzi hogy helyesen lettek-e az új játékhoz szükséges változók létrehozva illetve a mezők helyes kitöltését ellenőrzi

* ***GyorsulasLoadTest***: a fájlból való betöltést teszteli
* ***GyorsulasNewGameAfterLoadTest***: hasonló a ***GyorsulasNewGameTest***-hez csak itt még mindenek előtt betöltünk egy már meglévő játékállást
* ***GyorsulasTimerTickTest***: teszteli a ez idő múlását, az üzemanyaglerakást illetve a játékidő gyorsítását
* ***GyorsulasProgressGameTest***: teszteli az üzemanyagok mozgatásának helyességét
* ***GyorsulasMoveTest\_Left*** illetve ***GyorsulasMoveTest\_Right***: a játékos bal és jobb irányú mozgatásait tesztelik
* ***GyorsulasGameEndTest***: a játék végére érésének helyes észlelését ellenőrzi
* ***GyorsulasPauseTest***: a játék szüneteltetésének funkcionalitását teszteli
* ***GyorsulasMoveOnTopOfAFuel\_Left*** és ***GyorsulasMoveOnTopOfAFuel\_Right***: megnézik hogy helyesen van-e lekezelve mikor a játékos egy mellette levő üzemanyagcellára lép rá