## A FELADAT CÍME:

WordFinderGameNuxtJS

## A FELADAT RÖVID ISMERTETÉSE

Asztali számítógépen és mobil telefonon is egyarán használható szókereső játék. Egy megadott fájlból (JSON)-ból kiolvassuk a szavakat, amiből egyet a felhasználónak el kell találni. Addig kell új szavakat keresni a listában amíg nem talált egyet a játékos a listából. Megszámolja a szóban csak az egyszer előforduló betűket, ezek száma lesz a szóhoz tartozó pontszám. A szót és a pontszámot elmenti egy toplistába, amit ki lehet listázni, meg lehet nézni. Egy szó csak egyszer kerülhet a toplistába, nincs különbség a kis és a nagybetűk között.

## TELEPÍTÉS

- Sem a tömörített fájl, amit a levélben csatolok sem pedig a privát git repository - részletek a használt eszközök szekcióban - nem tartalmazza a 'node_modules' könyvtárat. Ezért kicsomagolás után, ha van NVM és node telepítve (NVM 0.33.0; NPM 12.16 vagy nagyobb) akkor a következőket kell futtatni a 'WordFinderGameNuxtJS' könyvtáron belül:
    - Npm install
        - Telepíti a package.json-ban lévő függőségeket
    - Npm run dev
        - Futtatja fejlesztői módban az alkalmazást
        - Server és egyéb paraméterek nincsenek beállítva

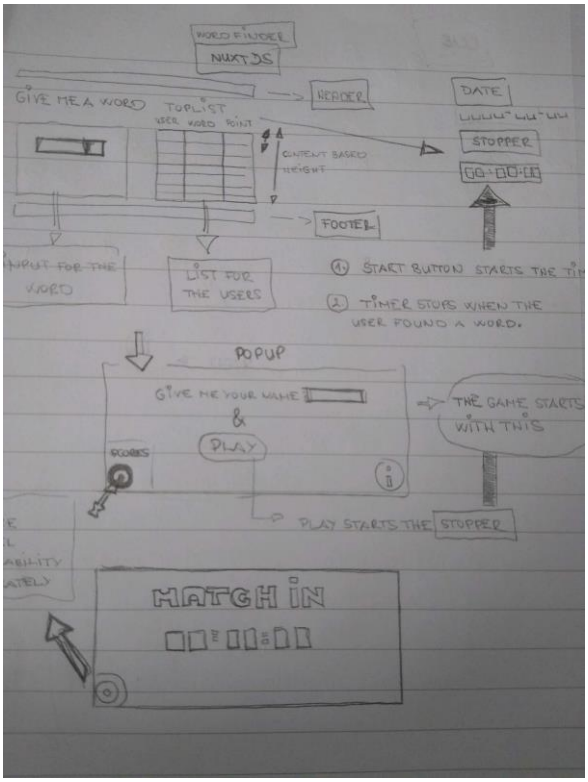
## ELKÉPZELÉS

A kapott specifikáció és email-es javaslatok alapján először papíron majd számítógépen próbáltam egy olyan grafikai megjelenítésre törekedni, ami felhasználóbarát és intuitív. Miután a FronEnd-es keretrendszerek közül a Bootstrap-el volt eddig a legtöbb tapasztalatom ezért mindenképpen szerettem volna azt használni. Így a Nuxt.JS-el is kompatibilis Bootstrap-Vue-ra esett a választásom.

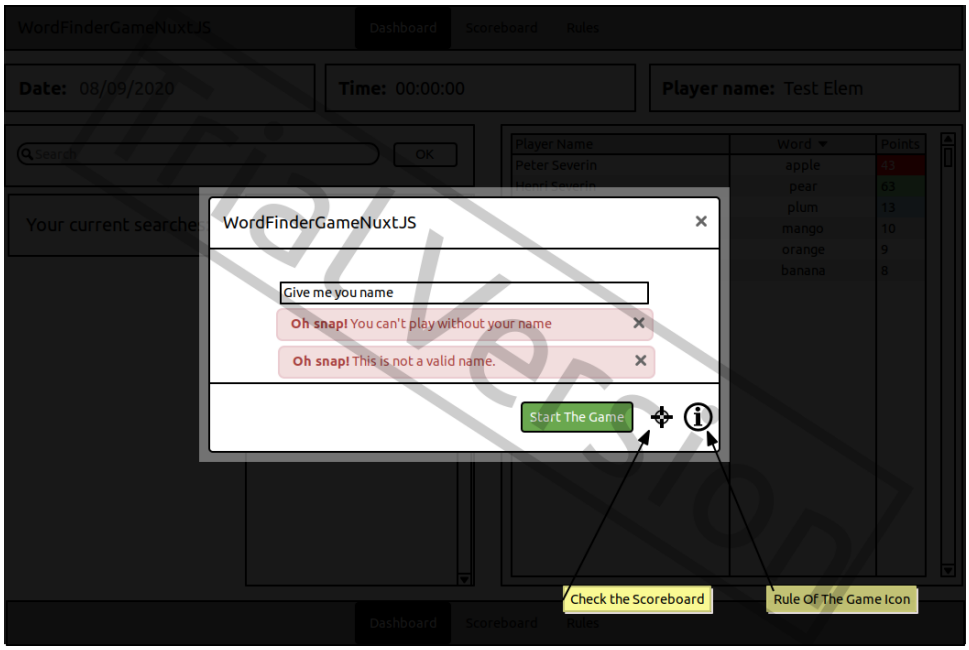
A wireframe-ek elkészítésénél egy fontos kritérium volt, hogy olyan eszközzel dolgozzak, ami ismeri a Bootstrap eszköztárát és Bootstrap kinézetű blokkokat tudok vele készíteni. Így a Wireframesketcher mellett döntöttem.

Mivel a kapott specifikáció sem megjelenítésről, sem pedig a játék logikájáról és menetéről nem adott részletes leírást ezért próbáltam kreatívan megközelíteni a kérdéses részeket miközben figyelembe vettem a rendelkezésre álló időt és a feltételeket.

## PAPÍRVÁZLAT



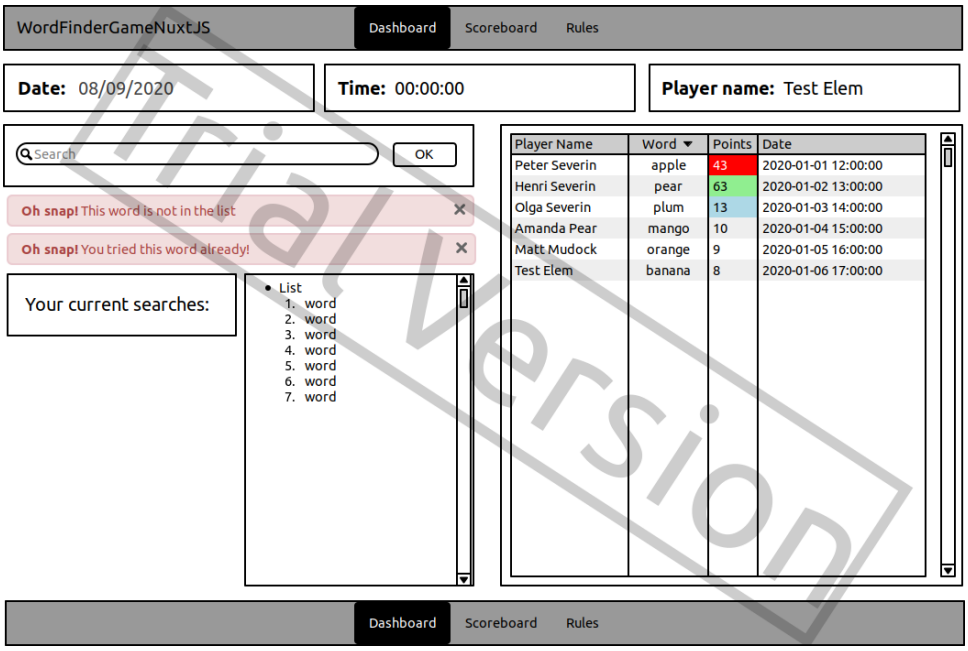

REGISZTRÁCIÓ & JÁTÉK




*Jellemzők:*

- *Automatikusan Felugró ablak, ami nem megkerülhető! Játékhoz ki kell tölteni!!*
- *Gombok: Játék kezdés, Játékszabályok, Toplista*

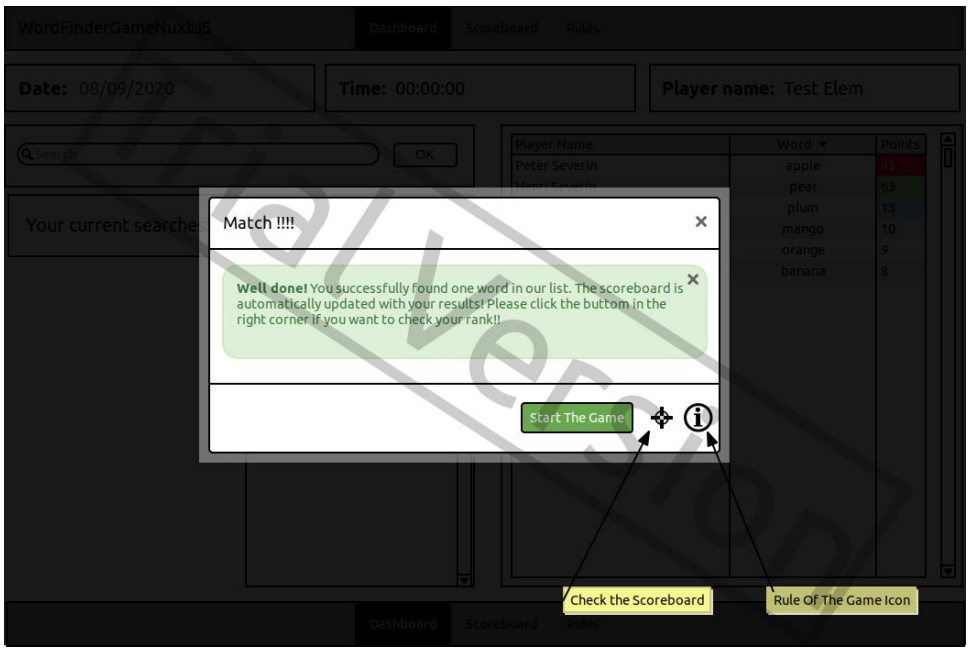
## VEZÉRLŐPULT




*Jellemzők:*

- *Logó*
- *Navigációs menü alul fölül*
- *Aktuális dátum, Idő/Stopper, Játékos neve*
- *Kereső mező, korábban már használt szavak, amik nem találtak lista*
- *Aktuális Toplista*

## TALÁLT ÉS NYERT!



*Jellemzők:*

- *Automatikusan Felugró ablak, ha a felhasználó találta egy szót a listából*
- *Gombok: Új Játék kezdés, Játékszabályok, Toplista*
- *Gratulációs üzenet*

JÁTÉKSZABÁLYOK:


*Jellemzők:*

- *Automatikusan Felugró ablak, a játékszabályokkal*
- *Gombok: Új Játék kezdés, Játékszabályok, Toplista*

PONT TÁBLA:

*Jellemzők:*

- *Automatikusan Felugró ablak, a korábbi játékosok eredményeivel*
- *Gombok: Új játék kezdés, Játékszabályok, Toplista*

## PROBLÉMÁK, DÖNTÉSEK ÉS MEGOLDÁSOK:

*Az adatok beolvasása milyen módon történjen?*

Az ötletelés során az első és legkérdésesebb pont a JSON file beolvasása volt. Ugyan volt több megoldási ötletem erre, hogy ezt, hogy lehetne megoldani, de a leírásban félreérthető node és npm kritériumok miatt bizonytalan voltam, hogy melyik megoldás irányába tudok elindulni.
Az ötleteimen felül a https://content.nuxtjs.org elég szimpatikus volt és a tőletek kapott email-es visszajelzés után úgy döntöttem, hogy a szavakat tartalmazó JSON beolvasására ezt fogom használni.

*Adatok kiírása táblázathoz. Hova és hogyan?*

Az első ötletem a 'sessionStoarge'-volt. Azonban a 'sessionStorage' ugyanazon browser két füle között már sajnos nem él ezért a 'localStorage'-fele terelődött inkább a figyelmem. Ez azért is előnyösebb mert új játék indítása után az adatok még mindig elérhetőek maradnának és nincs sem fülekre, sem pedig példányokra korlátozva. A harmadik ötletem a fájlba írás volt. Azonban azt nem tudtam, hogy erre milyen megoldást tudok találni. A dokumentáció alapján a *nuxt/content* képes visszaírni fájlba a módosításokat, de példákat sajnos csak in-line editorral egybekötve találtam, így végül inkább a 'localStorage' mellett döntöttem, bár később érdemes lenne még pár kört futni ebbe az irányba.

*Adatok megosztása komponensek között. Kód strukturálás.*

Habár Vue.js-el körülbelül egy hónap-ot foglalkoztam, és az adatok megosztása a komponensek között egy elég nagy kérdés volt. Az Angular-os tapasztalatom sokat segített nem csak a tájékozódásban, de a tervezésben is egyaránt. Itt fontosnak tartanám megemlíteni, hogy felhasználónév adatot a vezérlőpulttal célszerű lenne az 'Event Bus design pattern' használatával megoldani. Erről van is egy komment a kódban, azonban idő miatt ezzel nem foglalkoztam tovább.

## FEJLESZTÉSI KÖRNYEZET, PROGRAM VERZIÓK

- Ubuntu 18.04
- NVM 0.33.0
- NPM 12.16 vagy nagyobb
- Nuxt.js @ v2.14.4
- @nuxt/content ^1.5.0
    - Függőségek:
        - Bootstrap ^4.5.0
        - bootstrap-vue ^2.15.0
        - Nuxt ^2.14.0
        - nuxt-vuex-localstorage ^1.2.7"
        - vue-crono ^2.0.9
        - vue-underscore ^0.1.4

### HASZNÁL ESZKÖZÖK:

- Webstorm (IDE)
- Kanbanflow – Feladatok menedzselése (kérésre a kanban board-hoz tudok meghívót küldeni, a log pdf-ben csatolva)
- Git és Github (A repositoy az privát kérésre meg tudom osztani)

### TOVÁBBFEJLESZTÉSI LEHETŐSSÉGEK:

A rendelkezésre álló feltételek és idő alapján próbáltam a követelményeket úgy megvalósítani, hogy később, legyen lehetőség a továbbfejlesztésre. Az alábbi funkciók mindenképpen javítanák a játék élvezhetőségét.

- Megtalált szavak eltávolítása a JSON-ból.
    - Egy szó csak egyszer kerül a toplistába de szerintem célszerűbb lenne hosszabb távon eltávolítani a megtalált szavakat a listából ezzel ugyanaz az eredmény reprodukálható, de így még egyszer nem is lehet rákeresni a szóra. Viszont így kellene a játékba egy plusz logika, hogy ha csak egy szó maradt a listában generáljunk újakat, de ez sok egyéb dolgot is felvet.
- Ha játék nem az első szó megtalálása után érne véget, hanem lenne benne valamilyen extra limit.
    - Korlátozott idő (például: maximum 3 perc a játékidő)
    - Nem egy szót, hanem 3 vagy 4 szót kellene megtalálni
- Különböző témakörökben szerepelnének szavak. A játékos a neve megadásával egy időben megadná azt a kategóriát (például: autók) amiből szeretne szavakat kapni.
    - Ennek függvényében kellene bővíteni és betölteni a szavakat tartalmazó JSON fájlokat.
- Lehetne szinteket építeni a játékba.
- A program adhatna segítséget, hogy milyen irányba kellene gondolkozni. (Pl. Melyik amerikai autógyártó cégnek volt mérnöke Galamb József?)
- Játékos és a játék fontos adatainak tárolása szerveren.
- Az elkészített design alapján a felugró ablakok funkcióit, valamint a vezérlőpultot mindenképpen külön akartam kezelni, de egy esetleges továbbfejlesztés részeként a vezérlőpultot célszerű lenne tovább bontan: ‘Timer’, ‘Date’, ‘Name’,’Search’ és modulokra.
- A tervezés fázisában a ‘Timer-t’ egy rendes stopper-nek gondoltam, ami lejárhat, azonban az idő miatt ezt elvetettem amúgy az óra sem volt a feladat része eredetileg.

### ISMERT HIÁNYOSSÁGOK:

- Tesztelés során észrevettem, hogy van olyan eset, hogy a 'regisztrációs’ ablakot kitöltés nélkül meg lehet kerülni (ezt tesztelni kell mi ez az eset)
- Mobil felületen igazítani a felugró ablakokban lévő gombok kinézetén