# ECHIPA FRONT-END:

Teșu Cristina - ScrumMaster

Teodorescu Alexandru

Chipăruș Denis

Bîrzu Alexandra

Cîrstean Vlad

Păduraru Andra - Elena

Iordache Viorel

Corban Cristian

Țocu Andrei

Păduraru Teodor

Dăscălița Gheorghiță

Zbanț Radu

Hugeanu Felix

In procesul de slelectare al tehnologiilor ne-am concentrat pe doua criterii importante:

* Limitarea tehnologiilor folosite la un numar cat mai mic: deoarece orice tehnologie noua ar insemna invatarea ei de catre toti membrii echipei intr-un timp limitat
* Avantajele pe care le ofera respectivele tehnologii

Astfel procesul de alegere a insemnat balansarea celor doua pentru a obtine cel mai bun raport intre functionalitate si ease of use.

# TEHNOLOGII PE CARE LE-AM LUAT ÎN CONSIDERARE :

## ANGULAR

|  |  |
| --- | --- |
| **PROS** | **CONS** |
| Two-way data binding (View-ul și Model-ul se sincronizează continuu) | Performance (aplicațiile dinamice pot rula încet uneori) |
| Directives (permit crearea de conținut dinamic și bogat cu HTML) | Steep learning curve (destul de greu de învățat la nivel înalt) |
| Dependency injection (dependențele se separă de componente, făcându-le mai ușor de reutilizat, de organizat și de testat) |  |
| Community (comunitate mare, mai multe materiale și trainning-uri) |  |
| Google Long-Term Support |  |

## REACT

|  |  |
| --- | --- |
| **PROS** | **CONS** |
| Virtual DOM (îmbunătățește experiența utilizatorului și crește viteza de dezvoltare a aplicației) | High pace of development (mediul de lucru se schimbă constant și dezvoltatorii trebuie să învețe noi metode de lucru) |
| Permission to reuse React components (scurtează timpul de dezvoltare a aplicației) | Poor documentation |
| One-direction data flow (crește stabilitatea codului - modificarea elementelor copil nu afectează părintele) | JSX’s complexity (extensia JSX ce permite combinarea codului HTML cu cod JavaScript este destul de greu de învățat) |
| Open-source (o comunitate în continuă creștere) |  |

## REACT NATIVE (for Android and iOS)

|  |  |
| --- | --- |
| **PROS** | **CONS** |
| React Native uses JavaScript ( crește performanța ) | Instability, compatibility issues, and errors |
| React Native contains all ReactJS features, aimed at improving UI | Poor documentation |
| One-direction data flow (crește stabilitatea codului - modificarea elementelor copil nu afectează părintele) | Lagging SDK Updates ( când iOS sau Android lansează update-uri pentru SDK-urile lor, React Native se sincronizează greu cu aceste update-uri) |
| Open-source (o comunitate în continuă creștere) |  |

# TEHNOLOGII ALESE:

## VUE

|  |  |
| --- | --- |
| **PROS** | **CONS** |
| Size (ocupă puțin spațiu) | Lack of some common solid plugins/components |
| Ease of integration | It’s envolving fast (multe exemple de pe net pot fi outdated) |
| Scalability and versatility |  |
| Readability for the perfectionist (cod foarte ușor de citit, modificat și întreținut) |  |
| Rapid pentru proiecte de dimensiuni mai mici |  |

Vom folosi Vue Js ca baza pentru aplicatia noastra. Am ales acest framework din urmatoarele motive:

* Aplicatia va fi impartita in componente refolosibile. Acest lucru va eficientiza dezvoltarea, testarea, problemele vor fi isolate la nivel de componenta si taskurile vor fi usor de definit si distribuit
* Acest framework ne ofera posibilitatea de a updata usor si efficient interfata in momentul in care se modifica datele
* Folosind pentru toata aplicatia un framework, totul va fi structurat de la bun inceput
* Foarte light: 30kb
* Simplu de invatat (vanilla html, css si javascript) dar cu posibilitatea de a folosi Jsx daca se va dovedi necesar
* Este un framework modern folosit in industrie
* Ofera native add-ons: vue-router si vuex

De asemenea pentru un development efficient sunt necesare unele add-on-uri:

* Axios: recomandat de comunitatea vue pentru a comunica cu un API. De asemenea, pe langa faptul ca ofera toate functionalitatile de care avem nevoie, acesta este un Http Client bazat pe Promise, prin urmare poate fi folosit cu Async/Await, acesta fiind motivul pentru care l-am ales.
* Vue-router: fiind vorba de o aplicatie single page, url-urile nu vor corespunde cu ceea ce este afisat si va fi imposibila atat distribuirea de link-uri cat si folosirea de bookmark-uri. Din acest motiv vom folosi o extenie care ofera posibilitatea de a face mapping intre componente si url
* Vuex: consideram ca cel mai eficienta este sa avem un singleton care face management la state-ul tuturor componentelor
* Babel: vom scrie doar cod modern (ES6+) si vrem ca aplicatia sa fie compatibila si pe browsere mai vechi si dispositive mobile
* Jest: pentru a face unit-test componentelor vitale
* ESlint + Prettier: pentru a semnala potentiale erori si probleme

CSS:

* CSS-preprocessors (Sass, Less etc): nu consideram necesare, la momentul actual, functionalitatile ce sunt oferite de aceste tehnologii, dar suntem pregatiti sa adoptam Sass in caz ca se va dovedi necesar ulterior

Css-Frameworks:

* Alegerea noastra este Bootstrap: este un framework popular cu multa documentatie si are foarte multe componente care pot fi folosite

Alte optiuni de Css-Frameworks:

* Bulma: este light si css-only, dar in acelasi timp nu are la fel de multe optiuni ca bootstrap
* Materialize: stilul acestui framework nu e pe gustul tuturor
* Nothing: desi ne ofera cea mai multa customizabilitate, am considerat ca e mai eifcient dar si mai sigur sa folosim un framework care deja a fost testat de mii de utilizatori.