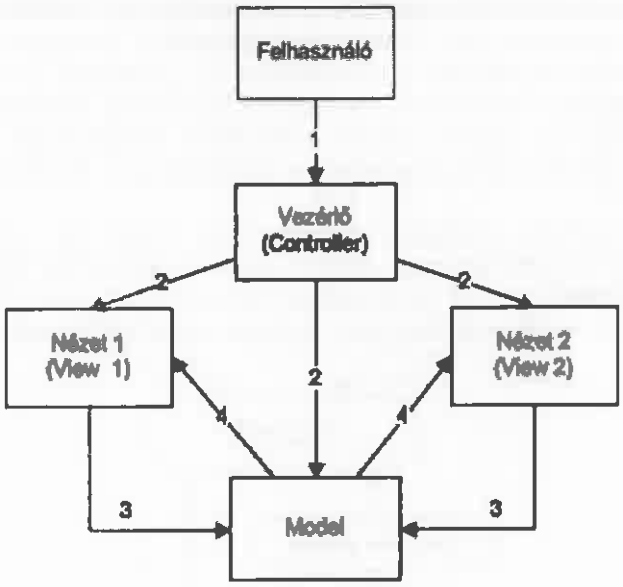
Az ábrán látható rendszer felépítéséből látható, hogy lehetővé teszi a redundáns működést, ami elengedhetetlen az alkalmazás zökkenőmentes eléréséhez. A felhasználók a szervezeti intranetes proxihoz (IntranetProxy) kapcsolódnak, amin keresztül érik el a portál rendszer elé illesztett Cisco terhelés elosztót (LoadBalancer, LB), ami irányítja a bejövő kéréseket az elérhetőség függvényében az Apache web kiszolgálókhoz (WEB01, WEB01-02). Ezek jelen esetben reverse proxy-ként funkcionálnak, elrejtve a mögöttük lévő rendszer karakterisztikáját. A hívás ezután a Mono-Server4 alkalmazás kiszolgálót futtató APP01 -02-nek adódik át, amiken a avatarOPERA belső portálok üzemelnek, melyek mindegyike eléri az relációs adatbázis-kezelő (MySQL), levelező kiszolgálót (Mail Server)

Az alkalmazott technológiák a back-end fejlesztés során .NET környezet elsődlegesen a C# nyelven.
A front-end fejlesztés során ASP.NET s (razor C) illetve MVC4, mert az alap szabványokhoz közeli technológiákat részesültek előnyben.

A követelmények alapján nagyon hangsúlyos, hogy a megvalósítandó rendszer platform független legyen előtérben az adatok és információk biztonságával. Ennek ellenére, alacsony teljesítményű hardvereken is a lehető legrövidebb válaszidőkkel fusson. A .NET technológia a mono-projektnek köszönhetően linux-os, Mac és Windows XP, 7 vagy újabb operációs rendszerek egyaránt jól és hatékonyan futtatható. A fejlesztés elsődleges kelléke az integrált fejlesztői környezet (IDE). A Visual Studio 2012, 2013 Express és SharpDevelop C# open source, ám teljes funkcionalitású eszközök. Az implementáció során a három alkalmazás réteg: felhasználói felület, alkalmazás logika, SOAP, adatbázis kezelés.

A programozási nyelv kiválasztása és az azon belüli technológia meghatározása után a következő lépés az alkalmazáson belüli architekturális minta megválasztása. Ebben az esetben a Model-View-Controller (Modell-Nézet-Vezérlő, MVC) került felhasználásra, mert korszerű és a MVVC maga is e köré a minta köré épül. A Modell a valóság egy darabja, amit egy vagy több osztály reprezentál. A Nézet jeleníti meg a tartalmakat, amit a Modell biztosít.

A fejlesztői környezet kiválasztása után a következő legfontosabb az adatbázis-kezelő kiválasztása.



Sok nagy múltra visszatekintő robosztus adatbázis kezelő létezik a piacon, amelyek között már régóta találhatunk megbízható open source változatokat is. A kiválasztásánál fő szempont, a megbízhatóság, a sebesség, az erőforrás igény, könnyű és gyors adatmentés. Szempont továbbá, hogy hibatűrőnek kell lennie. Ennek alapján a nagy múltra visszatekintő ORACLE robosztus és nagy terhelhetőség MySQL 5 adatbázis-kezelőjét és a könnyen telepíthető, nem túl nagy erőforrás igényű és megkötésektől mentes, valamint UTF-8 támogatással működő SQLite adatbázis kezelő került kiválasztásra, ami az alkalmazási területek széles skáláján már bizonyított. Megtalálhatjuk vírusirtókban ugyan úgy, mint Windows CE, vagy Android környezetben is megállja a helyét.