**Relational, lebo:**

Pros:

* **Consistency** means that any read request will return the most recent write. (ready vracajú najaktuálnejšie writy)
* **Availability** means that the non-responding node must respond in a reasonable amount of time. (neodpovedajúci uzol musí odpovedať v rozumnom čase; beriem to tak že databáza musí odpovedať v rozumnom čase)

Cons:

* NIE **Partition Tolerance** means that the system will continue to operate despite network or node failures. ALE TO NAM NEVADI (systém pokračuje aj keď dôjde k výpadku siete alebo nejakému zlyhaniu –výpadku uzlu-)

**WebApp, lebo:**

* povodne mala len stranka byt + skor sa dostaneme k ludom, nemusi instalvoat, updaty nemusime ludom nutit atd.

**Blazor Server-side, lebo:**

Pros:

* menej veci na sťahovanie
* bezpecnejsie lebo kod na serveru
* dá sa debugovať pekne
* vraj 20\_000 ludi naraz sa moze napojit bez toho aby sa spomalila appka, v nasom pripade to je v pohode
* Blazor server-side pre-renders HTML content before it is sent to the client’s browser. This makes it search-engine friendly, and there is no perceivable start-up time. (prerenderuváva HTML predtým než pošle klientovi do browseru, a preto je search engine friendly)
* Blazor server-side apps will work on older browsers as there is no requirement for Web Assembly, only HTML and JavaScript. (mal by fungovať i na starších browseroch lebo nepotrebuje WebAssembly, len HTML a JS)

Cons:

* offline mod sice NIE, ALE nám netreba
* Doesn’t work well in high latency environments (oneskorenie ?)
* Heavy resource demand on the server (asi preto lebo vždy zo serveru veci ťahá tak že ho zaťažuje)

- A TIE CONS NAM NEVADIA

+ C# už niečo viem zatiaľ čo JS nič extra