Tillgänglighet
Identifiera potentiella failures:

1. För många anslutna → server slö → värsta fall så kraschar den
2. Power outage → server dör
3. Database failures naturlig/malevolent → krasch
4. Incompatibility with browser/parsley unavailable to user - alla webbläsare ej klarar av att läsa koden
5. Outdated server software/parsley unavailable

Risk response tillgänglighet:

0. Logga ex för att veta när något går fel
1. Flera servrar, limit client calls error communications
2. Fysisk separata servers, backup ström generator
3. Backup db, SQL injections, Intelligent db manegment
4. Monthly check, don't use advanced/new tech
5. Continuous support and development

Utifrån de som står ovanför har vi kommit fram till active-passive pattern är mest lämplig i och med att active-active skulle skapa för mycket redundans.

Säkerhet
Säkerhetsrisker:

1. Användare får tillgång till data utanför ens behörighet
2. Extern aktör får tillgång till data/användaruppgifter
3. Användare får fysisk tillgång till databasen
4. Användare får adminrättigheter olovligt
5. Extern aktör får också oberättigad adminrättigheter
6.

Säkerhets respons:

1. Tydligt definiera behörighetsområden och implementerar
2. Kryptera kommunikation skydda mot SQL injections
3. Inget fysiskt gränssnitt till db, fysiska lås
4. Admin kan inte skapa ny admin, endast utvecklare skapar admin
5. Fysisk extra säkerhet för admin login
6. 1, 2, 4, 5 spara lösenord krypterat

Säkerhetslösning: mindre mönster för att lösa de ex hur man bygger db och koden.

Ingen controller → har ett annat ord för de
Istället för en server har vi flera

Prestation och trygghet - viktigt för att kunna använda de. Annars använder ingen de.

Säkerhet - för användare, permissions, vad kana användaren komma åt?
Tillgängligheten - för starka krav ex tillgänglig 24/7, alla kursdeltagare inne samtidigt. Men mer tillgänglighet än snabbhet.

1. Tillgänglighet -
2. Säkerhet - Användare ska inte kunna redigera andras projekt - med tanke på att den är skapad för skolelever.
3. Prestation - viktigt att alla kan använda tjänsten med acceptabel responstid.
   Klient-server löser de?

Slutsats:
Klient kopplar upp sig mot server1 om prestanda tiden inte överstiger de acceptabla, annars går vidare till server2. Ingen “mellanserver” för att minska single point of failure.

Valt active-passive pattern för att vi prioriterar tillgänglighet.
Mellan AP och AA → om en av s blir överbelastade så kommer alla klient som är anslutna att loggas ut när man använder AA → blir en viss tillgänglighets drawback.

Nästa möte: 27/2 13:15-17:00
Dina bokar sal