**Web-Queue-Worker/3 tier (SPA -> REST API -> DB)**

***System rejestracji pacjentów do przychodni***

Pytania:

- Jak bardzo złożony ma być system w kontekście funkcjonalności i widoków na frontendzie?

- Jak dużo wzorców projektowych powinno być użyte?

- Na czym się skupić? Na użyciu jak największej liczby wzorców projektowych czy do ich minimalizacji i dopasowaniu do wymagań biznesowych?

- Stosować kolejkę czy nie? Czy zastosowanie jej w tym przypadku jest biznesowo uzasadnione i daje większe korzyści niż architektura 3 tier, czyli:

**SPA->REST API->QUEUE->WORKER->DB** vs **SPA->REST API->DB**

Frontend w Angularze

Backend w Javie (non-blocking IO – użycie takiej funkcjonalności w serwerze servletów Tomcat)

Angular SPA wysyła requesty do Backendu w Javie przez REST API i Backend w Javie zapisuje dane w bazie danych SQL

Kolejka przy jakichś operacjach batchowych. Nie mam pomysłu jakie biznesowo operacje wymagałyby użycia kolejki tutaj… Może przy zapisywaniu się do przychodni może być bardzo duże obłożenie w pewnych godzinach, np. porannych? Wtedy operacja rejestracji musiałaby być kolejkowana. Na pewno trzeba by było również zastosować jakiegoś rodzaju lock (optimistic locking?) na daną godzinę i lekarza. Jeśli mielibyśmy workera, to on raczej nie byłby odpowiedzialny za obsługiwanie zapytań o dane?

Wzorce architektoniczne:

- automatyczne skalowanie wszerz w razie potrzeby?

- <https://learn.microsoft.com/en-us/azure/architecture/patterns/queue-based-load-leveling> (jeśli jest kolejka, to operacje które mogą się nawarstwiać mogłyby być wrzucane na kolejkę)

- <https://learn.microsoft.com/en-us/azure/architecture/patterns/materialized-view> (zmaterializowane widoki na bazie SQL)

- <https://learn.microsoft.com/en-us/azure/architecture/patterns/federated-identity> - Active Directory w Azure – OAuth2/OpenID, czyli Frontend pozyskuje token od Authentication Providera, a backend go waliduje poprzez klucze JWKS od Authentication Providera.

- Health check do aplikacji w Javie/frontendu w Angularze (czy dostępny)? - https://learn.microsoft.com/en-us/azure/architecture/patterns/health-endpoint-monitoring

- <https://learn.microsoft.com/en-us/azure/architecture/best-practices/api-design>

- można też zastosować jakiś cache, jeśli warto, tylko trzeba by znaleźć biznesowe uzasadnienie dla jego istnienia