Project risico's

Tijdens het maken van de projecten komen er natuurlijk altijd verschillende risico’s bij kijken. Hiervoor hebben we bepaalde onderwerpen op een rij gezet om ze even door te lopen.

**Transferability risico’s**

Dit project maakt deel uit van een pilot van Fontys dat later door andere projectgroepen wordt opgepakt en uitgebreid. De teams worden volledig vervangen per semester wat de nodige risico’s met zich meebrengt, aangezien alles opnieuw moet worden opgepakt en begrepen. Bij de nieuwe start-ups zal er dan ook steeds weer tijd verloren gaan bij het besteden van het begrijpen en later weer overdragen van de code.

Om dit risico te dekken wordt in de project backlog duidelijk gemaakt dat een feature altijd in volledigheid gedocumenteerd wordt. i.e. met comments en met officiële API documentatie met technieken als swagger. Het is daarom ook belangrijk dat er duidelijke readme’s worden gemaakt van eventueel nieuw toegepaste technieken en het opstarten van de services.

De architectuur die is vastgesteld in het architectuurdocument fungeert als basis van hoe het project zich interfacet met het IT-landschap van Fontys en ook met uitbreidingen van het systeem door volgende projectteams. Het is hiervoor belangrijk dat de architectuur bestanden zowel worden geüpload als de originele Visual paradigm bestanden en als de word bestanden. Zodat deze makkelijk aan te passen zijn in de toekomst.

**Privacy risico’s**

Personeelgegevens van Fontys worden streng bewaard in verband met privacy policy’s en wetgeving waar Fontys zich aan houdt. Contactgegevens van personeel worden niet zomaar vrijgegeven en we kunnen geen toegang krijgen tot een personeelsdatabase.

Dit risico kan gedekt worden door te kijken naar alternatieven zoals de active directory van de fontys email. Andere applicaties die in het IT-landschap van Fontys draaien maken daar gebruik van.

**Kosten riciso’s**

Licenties van Azure kunnen vrij duur zijn om als development team te gaan betalen van Fontys. Zodra het project in productie gaat hebben we daar minder last van, maar in de tussentijd moet met de blokeigenaar overlegd worden hoe de developmentkosten in rekening kunnen worden gebracht.

**Database as Single Point of Failure/Data loss**

Er is in het project gekozen om met microservices te werk te gaan. De database-laag wordt gezet in een eigen container om kosten van datamigratie en risico van data-loss te verminderen. Een risico dat hieraan hangt is dat de database een single point of failure wordt waardoor alle microservices uit de lucht vallen.