# Logical view

Frameworks

**Criteria**

Schaalbaarheid: In welke mate het geschikt is voor grote hoeveelheden aan gebruikers/rollen.

Onderhoud: Hoe vaak moet je code refactoren om up-to-date te blijven?

Documentatie: Goede documentatie maakt het makkelijker voor ontwikkelaars om uit te zoeken hoe het platform precies werkt.

Product maturity: Is het platform nog in de beginfase? Hoe actief is het team en hoe ziet de roadmap voor de toekomst eruit?

Team kennis: Kennis binnen het team betreft de technologie.

Ease of use: Hoe makkelijk is het te begrijpen/op te pakken?

Requirements: Heb je meerdere platforms nodig om het gehele systeem op te zetten? En zo ja, welke zijn dan geschikt?

**Front end**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Criteria | Weight | React | Angular | Vue 3 | Blazor | Asp.net | Ember |
| Schaalbaarheid | 8 | 9 | 8 | 7 | 5 | 6 | 6 |
| Onderhoud | 7 | 4 | 9 | 10 | 6 | 8 | 7 |
| Documentatie | 8 | 6 | 7 | 10 | 8 | 10 | 10 |
| Product maturity | 5 | 10 | 9 | 6 | 6 | 9 | 8 |
| Community | 8 | 9 | 8 | 9 | 6 | 8 | 7 |
| Team kennis | 6 | 4 | 5 | 7 | 2 | 9 | 2 |
| Ease of use | 9 | 6 | 4 | 8 | 8 | 7 | 6 |
| Requirements | 10(ja/nee) | 10 | 10 | 10 | 5? (QR code scanning maybe) | 0 | 10 |
| **Totaal** | 60 | 7.57 | 7.43 | 8.80 | 6.18 | 6.68 | 7.62 |

*React:* (Editor, The good and the bad of reactjs and react native., 2020),(Hettiarachchi, 2020)

*Angular:*

(Editor, The Good and the Bad of Angular Development, 2020), (Dan, 2019)

*Vue 3:* (Editor, The Good and the Bad of Vue.js Framework Programming, 2019),(Morrell, 2020),(vue-camera, 2019)

*Blazor:* (Guardrex, Introduction to ASP.NET core Blazor, 2020)*,* (Devathon, 2020)*,* (Guardrex, ASP.NET Core Blazor hosting models, 2020)

*Ember:* (EmberJS, n.d.)*,* (u/falkencreative, 2015)*,* (EvoCount, 2020)

*ASP.NET:* (dotne8dev, 2013)

**Conclusie:**

Op basis van de criteria hebben we gekozen om het frontend framework op te zetten met Vue. Mede omdat het een lichtgewicht en gemakkelijk te gebruiken framework is waar veel leden van het team al enige kennis van bezitten. De documentatie en onderhoudbaarheid speelde een grote rol omdat later andere teams ook moeten leren om met dit framework om te gaan als het project wordt overgedragen. Vue is meestal iets minder geschikt voor grote projecten, waaronder reactivity, complexity en het feit dat het nog niet vaak wordt gebruikt in grote projecten en de support daarvoor dus nog niet helemaal bekend is. Vue is het beste te gebruiken als bij het javascript framework vooral de focus ligt op de UI.

**Backend**

|  |  |  |  |
| --- | --- | --- | --- |
| Criteria | Weight | Java | .net core |
| Schaalbaarheid | **8** | **10** | **10** |
| Onderhoud | **7** | **6** | **7** |
| Documentatie | **8** | **6** | **9** |
| Team kennis | **5** | **7** | **7** |
| Community | **8** | **9** | **9** |
| Ease of use | **9** | **9** | **9** |
| Security | **10** | **6** | **9** |
| Integratie | **9** | **9** | **10** |
| **Totaal** | **64** | **7.80** | **8.90** |

(Java Security Overview, sd)

(Java vs .Net, sd)

(Questions tagged [c#], sd)

(others, 2018)

(docs.oracle.com, sd)

(docs.microsoft.com, sd)

(Oracle » JRE : Vulnerability Statistics, sd)

(Microsoft » Asp.net Core : Vulnerability Statistics, sd)

(Stegner, 2018)

(Questions tagged [java], sd)

(.NET Core and .NET 5 Support Policy, sd)

(Question: Really worth come from java to .net core?, 2019)

(Wierik, 2020)

(Stafford, 2020)

Conclusie:

Na inzicht in hoeveel kwetsbaarheden java heeft ten opzichte van asp.net core bleek dat de veiligheid van java een stuk lager lag dan gedacht. Ook de support van Microsoft voor ASP.NET in de vorm van updates en documentatie lag hoger. Java heeft grote updates 4 keer per jaar en Microsoft stuurt updates door wanneer nodig maar heeft een einddatum voor support van tevoren gesteld. De documentatie van ASP.NET core is overzichtelijker en ruimer. De integratie van andere soorten frameworks is mogelijk in allebei de frameworks, met de juiste libraries en API's kunnen allebei de formats samenwerken met bijna alles. Aangezien de opdrachtgever zelf in een Windows omgeving werkt ligt de integratiemogelijkheid daar iets hoger.

**Serverless computing**

Op basis van onderzoek naar artikelen over de voor- en nadelen van serverless computing in vergelijken met de wat traditionelere keuze voor puur werken met containers zijn deze resultaten gevonden. (Chan, 2018)

|  |  |
| --- | --- |
| Voordelen | Nadelen |
| Lagere serverkosten | Vendor lock-in |
| Automatisch opschalen | 5 minute time-out lange functies |
| Minder overhead infrastructuur development | Dependencies complexe applicaties moeilijk |
| Gaat goed samen met microservices | Langere responsetijd (verholpen met prewarming) |

Serverless architectuur is dus het meest ideaal bij applicaties die niet de hele tijd hoeven te runnen, als een applicatie lang stil staat maar op bepaalde tijden in moet springen, scheelt het veel kosten en development tijd om met serverless architectuur te werken.

**Welke provider?**

Met Azure hebben we een provider die aansluit op het software landschap van Fontys, het sluit ook aan op de .NET omgeving en bij de licenties die wij als studenten van school krijgen. (Microsoft Azure, n.d.)

Amazon heeft AWS Lambda. Dit is een tried-and-true serverless provider die ook gebruikt wordt door bedrijven die bezig zijn in hetzelfde domein, zoals door PostNL. (Amazon, 2020)

**Orm**

We hebben gekozen voor een ORM zodat we code first een database kunnen opzetten. Dit zorgt ervoor dat het aanpassen en opzetten van de database geheel vanuit backend code gedaan worden in plaats van database scripts. Dit scheelt een hoop werk en is overzichtelijk.

Om een ORM te kiezen hebben we gekeken naar ORMs die aansluiten op onze backend framework keuze .net core.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Criteria | Weight | Entity framework core | NHibernate | Dapper | LINQtoDB |
| Schaalbaarheid | **8** | 9 | 8 | 8 | 8 |
| Onderhoud | **7** | 9 | 7 | 4 | 7 |
| Documentatie | **8** | 10 | 10 | 10 | 6 |
| Team kennis | **5** | 5 | 6 | 0 | 0 |
| Community | **8** | 10 | 5 | 3 | 1 |
| Ease of use | **9** | 8 | 7 | 4 | 7 |
| Security | **10** | 8 | 8 | 8 | 8 |
| Product maturity | **6** | 10 | 10 | 7 | 6 |
| **Totaal** | **61** | **53.2** | **39.6** | **36.9** | **34.8** |

(C., 2019), (L., 2021), (Dapper 1.25.0, 2014), (Dapper, sd), (Newest "dapper" Questions, 2021), (Peres, Stackify, 2020)

**Database**

In de tabel hieronder zijn drie verschillende typen databases bekeken. Doordat we gekozen hebben .net entity framework valt NoSql al af. MSSQl en MySql staan zo goed als gelijk. MSSQL past beter bij .net entity framework dus hiervoor hebben wij gekozen.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Criteria | Weight | MSSQL | MySql | NoSql databases(MongoDB) |
| Schaalbaarheid | **8** | 10 | 10 | 8 |
| Onderhoud | **7** | 8 | 8 | 8 |
| Documentatie | **7** | 10 | 10 | 8 |
| Team kennis | **4** | 8 | 8 | 1 |
| Community | **5** | 10 | 10 | 7 |
| Ease of use | **5** | 10 | 10 | 6 |
| Security | **10** | 9 | 9 | 7 |
| Product maturity | **5** | 10 | 10 | 8 |
| **Totaal** | **51** | **46.8** | **46.8** | **35.5** |

(V., 2019), (Newest "sql-server" Questions, 2021), (What is NoSQL? NoSQL Databases Explained., 2018)

**Database hosting**

Om een goede keuze te maken voor het hosten van onze MSSQL database worden verschillende hosts vergeleken in de tabel hieronder.

|  |  |  |  |
| --- | --- | --- | --- |
| Criteria | Weight | Sql server selfhosting | Azure-sql |
| Onderhoud | **8** | 5 | 9 |
| Documentatie | **7** | 2 | 10 |
| Team kennis | **2** | 10 | 0 |
| Community | **5** | 3 | 8 |
| Ease of use | **5** | 5 | 7 |
| Security | **10** | 9 | 10 |
| Pricing | **10** | 10 | 5 |
| **Totaal** | **47** | **20.4** | **36.7** |

**Versiebeheer**

Git is de mees gebruikte manier van versiebeheer in de IT wereld. Git houdt de veranderingen aan een project bij waardoor er goed inzicht is in wat er per versie veranderd is en maakt het ook extra makkelijk om een rollback te doen. Je kunt via git zowel lokaal als op eens server draaien om het project bij te houden. School zelf heeft ook een plek om gitprojecten te bewaren en dat is een goed argument om het daadwerkelijk te gebruiken. Aangezien niet iedereen enorme ruimte op de harde schijf heeft is het handig om de versies van het project extern op te slaan. De voornaamste gratis manier is van school. Voor versiebeheer heeft school alleen git. Welk programma je gebruikt om git aan te spreken maakt niet heel veel uit, dat ligt aan persoonlijke voorkeur. De plaats waar git draait maakt iets meer uit maar daar is voor school 1 duidelijke optie voor.

Er zijn wel andere opties voor versiebeheer naast git, deze zijn echter niet optimaal. De voormalige populairste versiebeheer app was subversion. Er zitten echter een aantal nadelen aan subversion, waaronder de snelheid en foutbestendigheid zoals beschreven op backlog.com.

Dit maakt duidelijk dat git het programma is dat gebruikt moet worden.

(Git vs. SVN: Which version control system is right for you?, 2020)

(Noble Desktop, 2018)

**CI**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Criteria | Weight | Azure | Jenkins | Google | Amazon | Gitlab | TravicCI | CircleCI | Atlassian |
| Onderhoud | **7** | 9 | 9 | 7 | 5 | 9 |  |  |  |
| Documentatie | **9** | 9 | 8 | 8 | 7 | 8 |  |  |  |
| Team kennis | **7** | 5 | 4 | 1 | 1 | 5 |  |  |  |
| Community | **8** | 8 | 10 | 5 | 4 | 7 |  |  |  |
| Ease of use | **8** | 8 | 6 | 7 | 7 | 6 |  |  |  |
| **Totaal** |  | 7,9 | 7,5 | 5,7 | 4,9 | 7,0 |  |  |  |

*Azure:* (Editor, Azure Pipelines, sd)

*Jenkins:* (Pérez, 2017)

*Google:* (Agarwal, 2020)

*Amazon:* (Abhinav, 2020)

*GitLab:* (Jain, 2020)

**Containers**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Criteria | Weight | Docker | Azure container registry | Mesos |
| Documentatie | **8** | **10** | **10** | **9** |
| Community | **7** | **10** | **7** | **4** |
| Ease of use | **9** | **7** | **10** | **6** |
| Team kennis | **3** | **5** | **1** | **1** |
| Integratie | **8** | **8** | **8** | **5** |
| pricing | **10** | **10** | **3** | **10** |
| **Totaal** | **45** | **39.2** | **31.6** | **29.7** |

(docker questions, 2021),

(docs.docker, 2021)

(mesos documentation, 2021)

(build and push to azure container registry, 2019)

**Container orchestration**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Criteria | Weight | Kubernetes | Docker swarm | Amazon elastic | Azure kubernetes |
| Documentatie | **8** | **10** | **7** | **8** | **8** |
| Community | **7** | **10** | **2** | **3** | **2** |
| Ease of use | **9** | **5** | **6** | **5** | **5** |
| Team kennis | **3** | **2** | **1** | **1** | **1** |
| Integratie | **8** | **10** | **7** | **7** | **8** |
| **Totaal** | **35** | **28.1** | **18.3** | **18.9** | **19** |

(questions tagged [kubernetes], 2021)

(kubernetes documentation, sd)

(azure kubernetes service (AKS), sd)

**Code quality**

Sonarqube geeft de mogelijkheid om statische code analyse uit te voeren. Met Sonarqube kun je zien wat de kwaliteit van je code is, en het geeft suggesties van hoe je de code kan verbeteren om technical debt te voorkomen. (Sharma, 2015) Sonarqube richt zich niet alleen op het zoeken van bugs maar ook op de seven axes of quality: Spaghetti design, Comments, Coding rules, Duplicacy, Test-cases coverage, Potential bugs en Code complexity. (Nadeem, 2015)

**Frontend testen Sverre**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Criteria | Weight | Selenium | Cypress | TestCafe |
| Documentatie | **8** | 6 | 9 | 8 |
| Community | **7** | 10 | 7 | 6 |
| Ease of use | **9** | 5 | 5 | 7 |
| Team kennis | **3** | 2 | 4 | 3 |
| Integratie | **5** | 8 | 8 | 8 |
| Rapporten | **6** | 6 | 6 | 7 |
| **Totaal** | **38** | 23,9 | 24,2 | 25,1 |

*Selenium:* (Editor, 2021)

Cypress: (Editor, 2021)

*TestCafe:* (Jayaraj, 2020)

**Backend testen**

Om de backend te kunnen testen heb je 2 methodes nodig: Unit testen en integration tests.

Unit tests zijn kleine testjes die ingebouwd zijn in ASP.NET CORE die vooral onderdelen binnen klassen verifiëren en niet tussen de lagen door werken. Hier kunnen aparte methoden mee worden getest met verschillende soorten input en verschillende soorten verwachtingen.

Integration testen bekijken of het systeem in zijn geheel werkt als het wordt opgebouwd en spreken verschillende lagen aan, in complete gevallen van database tot front-end. Deze testen bekijken bijvoorbeeld of de onderlinge connecties goed verlopen. Voor integration tests kun je niet mocken.

Voor allebei deze soorten tests heeft ASP.NET CORE alle benodigde middelen al in het systeem zitten evenals in visual studio. Extra tools zijn dus niet nodig maar kunnen wel worden gebruikt.

(anderen, 2021)