

Research Document

Data opslag en storage



15 juni 2022

Justin van de laar

Inhoud

[Hoofdvraag 2](#_Toc106271185)

[Deelvragen 3](#_Toc106271186)

[Wat is de data die we nodig hebben voor de applicatie. 3](#_Toc106271187)

[Wat voor manieren zijn er voor data opslag en verwerking 4](#_Toc106271188)

[Data opslag different ways 4](#_Toc106271189)

[Data verwerking 9](#_Toc106271190)

[Issues die voorhand er al zijn in de applicatie. 10](#_Toc106271191)

[Ethischere regels voor een software developer 11](#_Toc106271192)

[Public 11](#_Toc106271193)

[Client and Employé 11](#_Toc106271194)

[Product 11](#_Toc106271195)

[Judgement 11](#_Toc106271196)

[Management 11](#_Toc106271197)

[Profession 11](#_Toc106271198)

[Colleagues 11](#_Toc106271199)

[SELF 12](#_Toc106271200)

[Leveren de ethische regels een probleem op voor de applicatie 13](#_Toc106271201)

[Product 13](#_Toc106271202)

[Public 13](#_Toc106271203)

[Conclusie 15](#_Toc106271204)

[Prototyping ideas. 15](#_Toc106271205)

[Bronnen 16](#_Toc106271206)

[Images 17](#_Toc106271207)

[Image 1.1 User table 17](#_Toc106271208)

[Image 1.2 Stats table 17](#_Toc106271209)

[Image 1.3 Structuur voor Distrubited database system 18](#_Toc106271210)

[Image 1.4 Verwachte Data structuur 19](#_Toc106271211)

# Hoofdvraag

Voor het ontwikkelen van mijn klikgame ben ik altijd tegen de vraag opgelopen van data storage. Wat ik voornamelijk hiermee bedoel is vooral de data van de user. Lijk de vraag natuurlijk ligt tot hoever heb je nodig en hoe store je dat het beste/veiligst.   
  
Dat zijn de vragen die ik wil beantwoorden met dit onderzoek. De main vraag is *hoe storen we data het veiligst/beste*. Hier komen natuurlijk een aantal sub vragen bij kijken. Zoals:

1. Wat is de data die we nodig hebben voor de applicatie
2. Wat voor verschillende manieren zijn er om data te opslaan/verwerken
3. Issues voorhand met de applicatie
4. Wat zijn ethische regels voor een software developer
5. Zijn er ethische regels die een probleem opleveren voor de applicatie

Dit zullen een aantal sub vragen die aan bod zullen komen. Uiteindelijk is het idee te kijken hoe andere applicaties het doen en ook een stukje te gaan prototypen. Tenslotte kan je maar op een manier kijken of je ideeën werken en dat is door het te testen.

# Deelvragen

## Wat is de data die we nodig hebben voor de applicatie.

Een issue waar ik altijd tegen aan liep en nog steeds heb is: Wat is de data die ik nodig heb voor de applicatie en waar ligt de grens? Waarom ik aangeef grens is vanwege het feit dat je hier werkt met User data. User data zal moeten worden behandeld als of het je meeste waardevolle bezit is want je wilt niet dat het op de straat wordt gelekt want dan zit je met grote problemen

Dus het idee is waar begin je dan? Wel als we kijken naar het onderzoek van AMC Ethics (2022) dan zijn er 8 principes waar we kunnen beginnen naar te kijken. De 8 principes zijn als volgt:

1. PUBLIC
2. CLIENT AND EMPLOYER
3. PRODUCT
4. JUDGEMENT
5. MANAGEMENT
6. PROFESSION
7. COLLEGAUS
8. SELF

Voor nu in het deel van het onderzoek is het nog belangrijk dat je weet wat de 8 dingen specifiek inhouden. Dit zal ik veder uitleggen onder het kopje van deel vraag 4. Voor nu is het alleen belangrijk om te kijken naar 2

Het onderzoek van AMC Ethics(2022) zegt voor nummer 2 dat: *Een software developer altijd in het interesse en manier werkt die het beste is voor de client and employee consent dit met de interesse van het publiek.* Waarom noem ik dit op is met de reden dat als User zou je niet al je data willen geven aan een game zoals we zien tegenwoordig.

Als we kijken naar Image 1.1 User table zie je dat het huidige ontwerp eigenlijk helemaal niet veel data verwacht van de user. Alles wat ik nodig heb of ter verwachting is eigenlijk een Email en een password. Want meer heb ik niet officieel nodig. Last\_location en inventory wordt behandel door de applicatie. Username is optioneel. Natuurlijk gebruik ik dat door de applicatie heen als display naam maar je zou der voor kunnen kiezen die het deel te maken van de Email.

Hier komt natuurlijk wel het risco mee dat persoonlijke opslag gehackt wordt. Ook kan je negatieve feedback krijgen als je de user geen eigen username laat kiezen of als de username toevallig als first and last name wordt gebruikt. Want sommige doen dat op hun email. Alleen de vraag blijft natuurlijk hoe sla je dit allemaal veilig op en heb je wel iets van opslag nodig?

## Wat voor manieren zijn er voor data opslag en verwerking

Dit stuk ga ik opdelen in 2 kleinere stukjes. Het eerste stuk zal voornamelijk gaan over data opslag en de verschillende manieren ervoor. Hier kijken we ook naar verschillende pro en cons hiervoor. Hetzelfde wil ik gaan doen voor data verwerking

Om het wat makkelijkere te houden pakken we het stukje login als voorbeeld en register. Waarom vraag je af? Het houdt het gemakkelijke als je weet dat beide stukken over hetzelfde gaan. Dit geeft ook een betere vergelijking voor later. Well Lets start met opslag mogelijkheden

### Data opslag different ways

Het is handig te weten voor we beginnen dat de applicatie een Webbased is. Wat ik hier mee wil zeggen is dat de applicatie een web app is en dat de opslag mogelijkheden zich voornamelijk daar op zullen richten.

#### Localstorage?

Om te beginnen is er een optie op de website van nature. In programmeer taal is dit localstorage en kan je zien als storage op de website zichzelf. Nou denk je wat moet je dan opslaan dat eventueel hier zou kunnen worden gebruikt. Wel denk er is aan als je inlogt of registert. Dan krijg je een token terug die je kan gebruiken door de applicatie of in dit geval je user gegevens waarmee de applicatie herkent hey dat ben jij.

Alleen er zijn natuurlijk wel wat pros and cons voor local storage. Wat ik aanraad is vooral ook zelf op onderzoek te gaan als je dit leest. Omdat je eigen applicatie misschien andere behoefte heeft. Voor nu ga ik data gebruiken van het onderzoek MacDonald (2022)

##### Localstorage Pros and Cons

Het artikel van MacDonald (2022) gaat voornamelijk over Localstorage vs Cloudstorage. Alleen der zijn toch wel een paar punten uit te pakken als pros. Om te beginnen is *localstorage als een van de makkelijkste om aan te passen kwa structuur en data.*

Wat ik voornamelijk hier wil zeggen is dat localstorage is vaak te vinden direct in de browser. Het heeft dus geen verbinding nodig naar iets externs of te zoeken naar een bepaalde file die toevallig net weg kan zijn of veranderd van locatie. Dit zorgt voor easy changes to eventueel data en kan local worden opgeslagen. Localstorage servers zullen ook makkelijker aan te passen zijn kwa geheugen en ram bijvoorbeeld als je het local draait.

*Your in control and no third party is required.*Wat je Vooral rond ziet in onderzoek is Cloud vs Local storage. Een ding wat daarin terug komt wat ik wel moet geven is het boven gegeven statement. Als je het nou gebruikt als server op data opslag op je website er komt niet zo snel iemand anders bij kijken of te wel een 3rd party zoals we dit noemen. Dit is natuurlijk een voordeel want jij bent volledig in control. Alleen dit voordeel is ook een snel nadeel

*Localstorage data is more open voor een XSS attack*Hoewel alles local is en je er toegang tot heb en makkelijke uit te breiden brengt het voornamelijk ook een gevaar mee. Om te beginnen moet ik aangeven dat ik nu voornamelijk over JS localstorage spreek. Dit is meer gevaarlijker als je dit als data opslag zou gebruiken omdat er sneller door een XSS tempering de data in verkeerde handen kan vallen. Dit is iets wat je natuurlijk niet wilt.

XSS is een cross side JavaScript attack normal. Het kan al iets zijn een link die lijkt als een link van de website waar je achter wel naar de website wordt omgeleid maar er een file tussen wordt gedaan waardoor de Attackers je data nu hebben.

Ook is het nodig te zeggen dat een localstorage server dit niet zo snel oplost hoort. Als je hier bijvoorbeeld de server op een high traffic setting set of niet met de goede firewall enz. beveiligt dan heb je alsnog dezelfde grote kans dat je data in de verkeerde handen komt

*Een localstorage server heeft vaak meer gespecialiseerd iT staf nodig*Dit is meer een issues dat komt mocht je niet de Localstorage van JS gebruiken maar een fysieke server. Waarom wel je moet zo server draaiend houden, want je wilt niet dat je applicatie online staat een opeens de server eruit flikkert en dat niemand je applicatie meer kan gebruiken.

Dit vraagt vaak gespecialiseerd IT personeel die hierin getraind is. Natuurlijk als je het 24/7 wilt up runnen moet je tools hebben die de performance and bandwith meten van je server om te kijken waar je problemen zitten. Meeste hebben hier simpel gewoon geen toegang toe. In beide het personeel en de tools dus wordt er al sneller voor de Cloud gekozen

#### Excel?

Een niet heel gek idee wat eigenlijk ook best vaak wordt gebruik in het veld is Excel als opslag. Je krijgt dan een file waar als je gegevens in staan. Der zijn al een paar voordelen en nadelen performance gericht die je al zou kunnen opnoemen. Alleen je moet eerst excel voor je zien als een tabel in word

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Username | Email | Password | Inventory\_id | Last\_location |
| Elise12 | [Elise@hotmail.com](mailto:Elise@hotmail.com) | 1235456 | 1 | 12 |
| Juju124 | [Justin555@live.nl](mailto:Justin555@live.nl) | 1243535 | 2 | 76 |
| Henk123 | [Henkie@outlook.com](mailto:Henkie@outlook.com) | 134253 | 3 | 12 |

Dit voorbeeld hier is hoe een Excel database eruit kan zien als je de tabel uit Image 1.1 User table zou pakken en om zou zetten van een SQL database naar Excel database. Natuurlijk is de vraag wat zijn pros and cons dan van een Excel database nou laat het me uitleggen.

##### Excel database Pros and Cons

Om te beginnen is een pro op het op moment dat een team memeber het team zal joinen en dus bijvoorbeel de database onder zich moet nemen. Dus dan krijg je het feit dat Excel heel herkenbaar is voor het meeste personeel.

Dit kan natuurlijk een heel groot voordeel meebrengen omdat de speficieke teammember zich niet hoeft te focussen op het programma. Wat ik hiermee bedoeld dat het geen tijd of amper tijd gaat kosten wat de team member hoeft te leren kwa programma. Zoals ze zeggen tijd = geld. Hiermee kan de developer eerder aan de slag en sneller met de database aan de slag.

*Data is easy om mee te rekenen en te visualiseren*Dit zijn er eigenlijk 2 in 1. Alleen het is wel een tool waar Excel voor gemaakt is. De opslag kan makkelijke worden gevisualiseerd en e.v.t mee worden gerekend. In geval van visualisatie is er niet echt een groot voordeel voor de applicatie. Het zit hem hier meer in het reken aspect.

Als we kijken naar Image 1.2 Stats table dan kan je zien dat er gebruikt worde gemaakt van stats in de game. In de image is momenteel alleen de user\_stats te zien. Alleen de game gebruikt ook de stats voor Monster en Items. Natuurlijk kan je denken zouden deze berekeningen niet in de FE moeten. Alleen de argumenten daarvoor kunnen worden gemaakt. Alleen hoe leuk zou het zijn als je dit doet in een apart Excel worksheet. Dit brengt natuurlijk dan ook weer problemen maar het zijn wel de mogelijkheden van Excel.

*Data organiseren gaat gemakkelijker met Excel*Naast het visualiseren en rekenen is Excel ook gemakkelijk voor het organiseren van data. Waarom zeg ik dit. Wel als een applicatie waar je constant data ophaalt als je een locatie bezoekt is het toch wel handig dat je data is georganiseerd.

Naast de benodigde login data is het ook handig voor de rest van de applicatie om georganiseerd om te gaan met de data. Dus geen monsters in monster\_stats proppen enz. Zoals eerder aangeven haalt de applicatie continu data op als users een locatie bezoeken. Dit kan van alleen een locatie tot een locatie en monster tot een locatie en monster en quest. Zo georganiseerd omgaan met de database is van important voor de applicatie

*Groote van de applicatie kan een probleem veroorzaken*Wat ik hier bedoel is eigenlijk vrij simpel stel voor je hebt 10 tabellen in je applicatie die nou dienen als worksheets in Excel. Nou als je een kleine applicatie hoeft hier geen probleem te komen want de Excel file zit gewoon op je server en groeit niet zo snel in size.

Alleen op moment dat je een applicatie hebt waar verwacht wordt dat 100-10000 users het gaan gebruiken komt er toch misschien een probleem aan bod. De size van je Excel File gaat groeien en snel ook. Als je een server hebt met veel opslag kan dit geen kwaad maar op het moment dat je server gelimiteerd is komen er problemen.

*Human error*Natuurlijk wordt de database groot en deels gedaan door de applicatie. Alleen zou er de tijd komen dat af en toe developer bezig zullen zijn met de spreadsheets. Hier is het gevaar groot voor Human error.

Fouten waar normaal aan gedacht kan worden is data verkeerd verwerken. Verkeerde data verwerken of juist verkeerd opslaan. Je zou denken dit kan toch snel worden verholpen met dingen als een backup van de spreadsheet of een 2de oog nadat iemand klaar is. Alleen je moet denken dit is weer tijd weg voor het maken van een backup en ook weer tijd weg van de 2de medewerker die ook aan iets anders had kunnen werken. Zoals al eerder aangegeven tijd = geld.

De volgende opties komen ook uit keuzes met behulp van het onderzoek van *(DBMS | Types of Databases – Javapoint, 2021)*

#### Centralized (Mysql) Database?

De huidige optie voor de applicatie. Je moet het nu zien dat we een centrale database hebben waar alle tabellen staan kwa data. Dus we hebben nu als voorbeeld de Excel spreadsheets in 1 plek onder 1 database. Dit brengt natuurlijk heel wat voordelen mee maar ook een heel wat nadelen. Als je zou kijken naar heden daagse database structuren is dit de meeste voorkomende structuur kwa databasen.

*De datastructuur is een van de bekendste bij developers* Ook als aangegeven bij Excel is dit een structuur die bekend is bij heel veel developers. Dit scheelt weer tijd want de structuur is bekend en ja als gezegd is tijd = geld. Wat wel een probleem op kan leveren is je tool die gebruikt hier. Dit met de reden dat er heden daags zoveel tools zijn voor database oplossingen dat de developer die joint hier even op moet inkijken

Het kan ook voorkomen dat de developer weet hoe de tool werkt en dat je een nieuw teammember hebt die direct kan instappen bij als team member. Wat natuurlijk weer heel veel scheelt op kosten en tijd en het proces verder helpt.

*Centralized database houdt het gemak van communicatie van cross-projects.*Stel voor het we breiden de game applicatie uit nu met een admin panel die eventueel users zouden kunnen bannen, locaties toevoegen, updaten en verwijderen. Dan wordt het hebben van een centralized database toch een stuk makkelijker dan wanneer je een Distributed database hebt. Waarom hoor ik je vragen.

Wel zie het zo je hebt nu een applicatie of twee die onder dezelfde database vallen. Als ik een locatie zou updaten op het admin panel is het natuurlijk wel nodig dat het spel ook de change zou kunnen zien. Dit gaat lastiger bij een Distributed database structuur met de reden omdat hier vaak services worden verdeeld. Hier kom ik nog wel later op terug

Voor nu is het goed te onthouden dat een centralized database het updaten van data makkelijkere terug kan geven bij meerdere projecten die met dezelfde data werken. Dit kan een huge voordeel zijn omdat je dan geen man kracht hoeft te besteden aan het verwerken van de data in alle projecten.

*Sql Queries kunnen open liggen voor Injects als ze niet goed worden behandeld.*Dat is right als we zouden gaan voor een MYSQL database dan is de kans groot dat je SQL query’s open staan voor SQL injects. Nou hoor ik je vragen wat is een SQL inject query dan? Denk is aan het feit als je inlogt wat de query dan wordt is

SELECT \* FROM USERS WHERE username=”Elise123” & password=”12345”

Nou stel je voor dat je in het tekstveld invult dat je username is: Elise123 OR 1=1 Het rode gedeelte is waar het gevaar kan zitten omdat je query dan opeens eruit komt te zien

SELECT \* FROM USERS WHERE username=”Elise123” OR 1=1 & password=”12345”

Sinds OR 1=1 is true alle user table rows worden terug geven en dit is een groot gevaar. Met de reden omdat iedereen dan opeens bij data van users kan. Hierin kan hele gevoelige data zitten als e-mails en telefoon nummers of zelfs adressen als je denkt aan applicaties als Amazon. Ik moet wel zeggen dat het minder snel hier werkt omdat je ook nog & password hebt. Alleen als het alleen op username ging dan was dit Inject gevoelig.

*Op het moment van data corruptie ben je ook heel de database kwijt.*Een voordeel van de Distributed database system is dat als data corrupte in de database komt dat je niet alles kwijt bent en de hele datastructuur offline moet. Dit is wel het geval vaak bij een Centralized database structuur.

Dus besides de SQL Inject query’s is er ook heel hard nodig dat er wordt gekeken naar een goede back-up structuur. Je wilt niet hebben dat er corrupt data in je database komt of dat hierdoor je hele data laag down moet om het te fixen. Dit zorg juist weer voor problemen want de website kan niet worden gebruikt en hier door loop je traffic mis. Wat kan zorgen voor interactie bijvoorbeeld of ad revenu van ads op de website.

Ook kost het weer geld voor de developers om de problemen te fixen. Nou is het IT veld een beter werkplek voor dit dan wanneer je de boekhouding doet op een verkopers bedrijf zeg maar.

#### Distributed (Mysql) Database?

We hebben het al kort gehad over het feit van een distributie software system of in dit geval de database structuur. Wat kan je verstaan onder een gedistribueerd software system is dat stel je voor we hebben de user en de monster en de items.

In een centralized database structuur zit dit allemaal onder dezelfde database structuur op een plek. In een Distributed structuur heb je meerdere database waarin elke verantwoordelijke is voor hun eigen stukje. Dus dan krijg je een database voor de locaties en de monsters en de items. Dit brengt natuurlijk heel wat voordelen met zich mee maar ook wat nadelen

*Iedereen is verantwoordelijke voor hun eigen stukje*Wat je vaak ziet in dit soort structuren dat je vaak verantwoordelijke bent voor een stukje van de structuur. Om dit visie te geven heb ik hier de volgende afbeelding in Image 1.3 Structuur voor Distrubited database system Hierin kan je een prima voorbeeld zien van wat er gebeurt op een locatie waar een monster gevecht is.

Hierin worden natuurlijk updates gedaan of monsters en items en inventorys en een user opgehaald. Alleen als er iets moet worden geupdate aan bijvoorbeeld de user kant dan krijg je er niks mee te maken als monster team. Dit in het einde zorgt ervoor dat er minder kans is op breken en bugs waardoor er minder downtime is voor de applicatie. Ook aan de andere kant als je werkt in het Monster team en je update iets heb je ook niks te maken met de andere teams.

Dit zorgt voor snellere updates en meer uitbreidingen maar zoals ook gezegd hoort dit te geven aan het feit van minder bugs en crashes waardoor je minder downtime hebt.

*Minder lost op data corrupt gegevens*Zoals al eerder aangegeven op het moment van corruptie ben je minder gegevens kwijt of heb je minder downtime. Dit met de reden omdat een stukje van de applicatie maar in dit geval geaffecteerd is i.p.v. de hele applicaties data structuur. Dit in ruil zorgt ook voor dat de website niet compleet down hoeft in het minst ergste geval. Natuurlijk kan bij corrupt gegevens nog altijd natuurlijk ervoor zorgen dat de hele data structuur down moet voor het updaten van een fix ervoor of het draaien van de de backup.

*Het kost minder tijd om je verantwoordelijke heden te leren*Zoals eerder aangegeven kost het ook veel minder tijd om de structuur van je verantwoordelijke services te leren. Waarom hoor ik je vragen. Nou nu hoef je een bepaald stukje van de applicatie te leren wat minder tijd kost

Nu als je in het monster team werkt dan hoef je alleen dat stuk van de applicatie te leren normaal zou je niks erbij hoeven te leren. In het geval van mijn applicatie zou het ook niet hoeven te voorkomen omdat je als services gewoon je data terug geeft aan de FE. Die verwerkt het dan op de benodigde manier en laat zien wat het moet zien of geeft data aan je terug die je moet opslaan.

*SQL Injects zijn ook hier nog een probleem*Zoals eerder uitgelegd bij Centralized systeem zijn ook hier nog SQL injects een probleem. Dit komt met de voorwaarde dat het gebeurt op het moment dat je de data niet goed of afhandelt of verwerkt. Het geval hier wel is dat op een moment van een breach er veel minder data naar buiten komt. De reden hiervoor is om dat er dan vaak een inject op de services, Dus de injects zijn een pro en con at the same time als vergeleken met de Centralized database structuur.

### Data verwerking

Als je zou kijken naar verschillende data verwerking tools dan ligt het voor kwa login een paar idees open. Ik ga adden tot het idee dat de applicatie nu een aparte FE wordt met een API BE erachter. De FE kan de API callen voor data en de data wordt opgehaald en terug gegeven. Alleen hoe beveilig je dit.

Voor een paar voorbeelden wil ik het onderzoek van Dizdar (2022) erbij pakken. Hun gaan in heel erg detail over API security maar ik wil een paar methodes gebruiken.

#### O2Auth

Een common methode om het voor login systemen in heden daagse tijd is een 02auth system. Kijk maar naar de grote bedrijven zoals Google en Facebook. Hun hebben een O2Auth system waarin je kan inloggen op andere websites door hun services. Je klikt daar meestal op een button en dan wordt je omgeleid.

Vervolgens Log je in of register je met hun services en volla. De applicatie heeft niks meer met je gegevens te maken of ja minder dan voorheen. Want next time vraag je gewoon aan google de gegevens en check of je die kloppen. Zo ja log ze in en geef ze een token die kunnen gebruiken voor de Api. Alleen de vraag hier nog blijft.

Stel je voor in mijn geval. Ik heb een username, email en password nodig om op te slaan. Want website geven vaak ook nog hun eigen login beschikbaar. Dit in het geval dat je ff niet een keer door een aparte services wil inloggen. Uit experience heb ik toch al gezien dat je meer binnen krijgt van bijvoorbeeld google dan je denkt. Dus op moment dat er een malicious file wordt geüpload die dat trackt dan kan het problemen opleveren.

Nou heb ik al ook gemerkt dat google hun eigen fail safe hebben hierin. Je moet specifiek zeggen van welk adres de data request mogen komen. Ook al doe je het in een test applicatie. Dit zorgt natuurlijk voor veel meer veiligheid omdat het simpel request blokt van adressen niet in de White list

#### JWT tokens

Wat je vaak ziet in API structuur is dat er gebruikt wordt van **J**son **W**eb **T**okens. Dit zijn tokens die vaak als een grote stuk tekst. dFhdf54FDgRER453fdsf. Dit kan een voorbeeld zijn van een token. Vaak worden die tokens ook nog een id en expire date gegeven.

Wat is het dan met JWT tokens dat het zo vaak gebruikt wordt. Het wordt namelijk gebruikt vaak in combinatie met andere Login of Register methodes zoals O2Auth. Hoe werkt de key dan. Je moet het zien als een sleutel die specifiek voor jou gemaakt is. Om het een beter idee te geven moet je denken alsof je woont in een appartementen complex.

De appartementen complex is je API en iedereen wilt er gebruik van maken en naar binnen naar hun appartement. Alleen je hebt heel veel huis deuren dus hoe weet je nou bij welke deur je naar binnen moet of het complex überhaupt mag betreden. Nou well daar komt je key kijken. Met de key kan je bij de security vragen of je hier langs mag en krijg je een antwoord. Dit gaat vaak in naam of request zoals 401.

## Issues die voorhand er al zijn in de applicatie.

Ik moet zeggen dat de applicatie al momenteel in een state dat het zou worden kunnen gereleaset. Dit komt natuurlijk niet zonder zijn issues en bugs en problemen. Ik zal er een paar opnoemen waarvan ik al weet dat het issues gaan veroorzaken.

*Links en paths naar files*Momenteel is de applicatie ontwikkelde in een fase waar in de direct host gestimuleerd is als virtual host. Voor Laravel applicaties ga je normaal naar: *Foldernaam/public/index*. Hier vindt je dan de applicatie. Alleen ik heb het ingesteld dat er een virutalhost is die op dat adres het ernaar toewijst. Dus in plaats van de aangegeven folder krijgen we: *avontuur.nl.*

Nu denk je wat voor issues kan dit veroorzaken wel het kan veroorzaken dat compleet de website breekt. Omdat als het niet goed wordt afgehandeld dat dan links kapot zijn of wijzen naar bestanden die niet bestaan. Hierdoor niks tonen of het scherm. Javascript gaan kapot en waarschijnlijk zijn er geen CSS styling. Dus het zijn kleine dingen die wel aandacht vragen.

*In de huidige staat is data onveilig*De primair reason dat dit onderzoek wordt gemaakt. Als ik nu de applicatie live zal gaan laten gaan zit het achter een database structuur die waarschijnlijk 1, 2, 3 te hacken is. Het project is natuurlijk heel lang een persoonlijk project geweest met zicht op deployment. Alleen ik kon de data structuur nooit echt goed beveiligen.

In long term is het ook een grote risk voor de hele applicatie. Als eerder uitgelegd draait de applicatie op een Centralized database structuur. De database is zeg maar het hart van de applicatie. Op het moment van corrupte data of er iets wordt gelekt en mensen met kwade intenties krijgen toegang tot de database dan kan de applicatie offline geheid van 99% van de tijd. Dit zijn voornamelijk 2 problemen die effect hebben op de applicatie.

## Ethischere regels voor een software developer

Zoals eerder uitgelegd zijn er een aantal ethische regels waar een software developer zich dient aan te houden. Ten slotte werken we nou met gevoelige data. Als je wilt weten welke 8 het zijn kan je terug kijken bij de eerste deel vraag maar ik zal ze kort hieronder toelichten.

### Public

Als software developer dien je altijd software te maken met de interesse van de public. Dus in dit geval als je kijkt naar de game wil je niet als consumeer te veel gegevens geven aan de applicatie.

### Client and Employé

Dit stukje zegt eigenlijk hetzelfde als het deel van public. Maar het richt zich meer dat je als software developer altijd de beste intrest voor de client en je employees kiest. Dus dat je meer richt op de consent van je client en employees met in gedachte het idee van de public interesse

Dus denk eraan dat als je hier als client aangeeft ik wil dat de game een quest systeem krijgt die wordt getrackt door de game heen. Dan ga je niet zeggen lukt niet of pak je de duurst optie. Nee je kijkt honest hoe je dat kan doen op de beste manier voor beide je medewerkers en de client.

### Product

Hier wordt verwacht dat het opgeleverde product de hoogste kwaliteit en standaards heeft die je kan bieden als software developer. Dus niet een half afgebekte buggy project opleveren aan je klant maar een werkende versie met eventuele testen om te kunnen aantonen dat je bepaalde keuzes ook daadwerlijk werken.

### Judgement

Een rare op de lijst maar het gebeurt toch. Als je ethische bezig wilt blijven als software developer dien je altijd te handelen en keuzes te maken van uit een independent mind set. Dit gaat voornamelijk ook met de gedachte dat er geen kwaadaardige mensen een schoen tussen de deur kunnen krijgen in je applicatie om het zo te zeggen. Dus altijd dien je als software developer keuzes te maken vanuit het perspectief van jezelf dat ook het beste is voor de klant en je medewerkers

### Management

Dit stukje focust zich meer op de leiders van een software developer team of aka de managers. Als manager van het project dien je ter alle tijden een Ethical approach te blijven gebruiken en te promoten. Dit met de reden dat je vaak als manager ook ziet dat jij er in gelooft dat rest ook volgt. Ook dien je te checken of je medewerkers zich wel houden aan de gemaakte Ethical afspraken.

### Profession

Vanuit het oog van de publieke interesse dient een software developer altijd zijn/haar professionele status te behouden. Wat ik hiermee bedoel is dat je altijd professioneel gedraagt en opdient tegen over het publieke interesse. Dus als je een klant die je het meeste haat op de wereld ga je niet nog een backdoor in de applicatie achterlaten vanwege personal redenen. Nee dan gedraag je ook professioneel in beide gesprekken met hem en het bouwen van de software.

### Colleagues

Het is misschien een beetje een rare op de list maar toch niet al te out of the box. Je dient altijd supportive en eerlijk te zijn voor je collega`s. Natuurlijk hoeft dit niet op alles 24/7 want liegen is soms goed voor het moraal. Maar op werk gebied zal het zo altijd moeten worden uitgeoefend. Dus als een collega iets maakt dat echt niet kan aanspreken en eerlijk erover zijn in plaats van te zeggen: Ah heb je goed gedaan we Mergen het. Dan er na het compleet te gaan reworken. Probeer hier ook op te letten of je het uit kan leggen aan je collega wat mis ging.

### SELF

Dit stukje is eigenlijke simpel te beschrijven het gaat ervan uit dat je als software developer een leven lang je skills blijft ontwikkelen. Het verwacht wel dat je inderdaad op professioneel gebied je skills blijft ontwikkelen met ook een oog op de ethische omgang.

Zoals je ziet zijn er heel wat onderwerpen te behandelen voor een ethische aanpak. Persoonlijk denk ik zelf nog wel op een aantal gebieden te kunnen ontwikkelen. Alleen leveren als je kijkt naar deze 8 eigenschappen uit het onderzoek van AMC Ethics (2022) een probleem voor de applicatie.

Hier wil ik vooral gaan kijken vanuit een oogpunt van me eigen skills. Natuurlijk kan je zeggen in het algemeen maar het project is tot nu eigenlijk alleen gedeveloped door mij met hulp van andere natuurlijk vandaar de keuze hiervoor.

## Leveren de ethische regels een probleem op voor de applicatie

Als je terug krijgt op de 8 eigenschappen dan zijn er wel wat problemen die optreden op ethische gebied. Wat hier wel een probleem is die optreed dat het voornamelijk een 1 mans team is. Dus dingen als collega`s en management komen minder snel aan de order dan wanneer ik een team zal werken. Toch wil ik een paar kopjes pakken en uitleggen waarom ik denk dat er problemen opkomen

### Product

Eerste wat ik wil aangeven is het product zelf. Hoewel ik echt impressed ben van sommige features en hoe ze eruit zijn gekomen. Ben ik toch bang dat de kwaliteit van het product toch erg laag ligt. Ik zal een paar problemen opnoemen waarvan ik denk.

*Centralized Database of de verwachtingen van een database*Het eerste probleem dat veel tijd kan kosten of problemen is de database. Hoewel mijn eigen idee was een Point to click adventure game waarvan die game draait op een singel database heb ik af en toe toch wel me twijfels gekregen. Dit zit hem voornamelijk in de security idd en problemen met de database.

Het is natuurlijk heel leuk en je zou nu makkelijk een andere applicatie er aan kunnen hangen en der zal weinig tot niks hoeven te worden aangepast. Alleen op het moment dat er iets gebeurt met de database ligt daadwerlijk ook de hele applicatie stil. Dit is nou eenmaal een min punt en zeker als ik het publieke zou maken met het gevaar van buiten.

Dit is natuurlijk wel te voorkomen maar geeft wel aan met een stukje kwaliteit. Ook zitten er nu nog geen written testen op de applicatie. Zo een stukje quality waarin je kan zeggen van hey dit werkt niet op release en dan wijst naar een test van het zou moeten werken. Ik point dit omdat het nog steeds een stukje quality is voor de applicatie.

Vandaar denk ik dat dit stukje toch nog aardig wat aandacht verdient. Ik kan ook nog wel even doorgaan met dingen die anders kunnen of beter maar ik hou het hier even op.

*Code is???????*Een stukje dat nog toevoegt aan de quality is de code. Wat bedoel ik hiermee te zeggen is dat de code als ik er nu op terug kijk erg outdated is. Ik heb er voornamelijk nu ook moeite mee het terug te lezen. Laat staan iemand die het team zou moeten joinen. Die persoon zou het waarschijnlijk ook een aardig tijd gaan kosten om de code te leren.

Alleen dat is niet het echte probleem. Als ik nu zo terug zou kijken op de code en je zou een bug of probleem krijgen krijg je vast een issue met de code. Waarom het is best aardig nestelt dus een bug of issue kan er voor zorgen dat heel veel code zou moeten worden herschreven en dit is natuurlijk eigenlijk niet kwaliteit. Waarom? Wel het zorgt voor extra werktijd en load die moet worden besteed aan het verwerken van de bugs. Deze tijd had ook kunnen worden besteed aan het uitbrengen van new features

### Public

Een kopje dat ik ook nog wil bekijken is de public interesse. Voornamelijk is dit project een zelf own developed project geweest. Dus ik denk wel is wat als ik dit nu zou releasen en mensen gaan het spelen wat dan. Wat is de feedback die ik kan verwachten hoe vinden het mensen het spel en toch weer dat stukje security.

Wat bedoel ik nou feedback krijg je wel en alle soorten en maten. Het is meer het stukje security waarvan ik me zorgen maak en ook het stukje data dat mensen moeten geven. Like waar ligt de grens voor het publiek.

*Security in de applicatie*Het eerste probleem is Security. Hoewel ik over me tijd verschillende dingen hebt geprobeerd te implementeren tegen dingen als XSS en SQL injections. Het is nou eenmaal een tijd terug dat ik het gedaan heb. Zoals je als software developer moet weten met tijd komen nieuwe en betere methodes voor zulk soort attacks. Het is van nature dat je probeer up to date te blijven op zulk soort veranderingen

Nu ik eraan denk is het een soort stukje ook voor Profession omdat je leert met preventie om te gaan op ethische niveau. Alleen ja het iets wat je moet leren en doen maar ook een angst voor mij als ik het nu op dit moment van typen online zal gooien.

*Interesse van de public*Een voordeel van iets waar je van probeer te leren en je ontwikkeld voor jezelf is dat je nooit echt met ander mans interesse rekening hoeft te houden. Alleen stiekem in me achterhoofd hield ik natuurlijk de gedachte van ik wil het multiplier supportive maken. Alleen met dat komt natuurlijk opeens een andere interesse aan bod en dingen die ik eigenlijk zou moeten gaan maken waar ik zelf nooit aan gedacht had.

Het ding hier meer is in hoever moet ik gaan afwijken van mijn origineel idee en zou ik daar klaar voor staan mocht ik dit ooit supportive maken voor de public. Er zijn hier nog wel wat Balance issues die aanpassingen nodig hebben alleen ja Balance is niet altijd makkelijke.

Ook komen met dit idee weer issues die de performance van de applicatie kunnen beïnvloeden het grootste probleem gaat hem zitten naar mijn gevoel in het combat system. Waarom? Wel het is nu gecodeerd naar een systeem waarin het loopt dat ieder jij of het monster dood is.

Alleen maakt dit niet uit maar denk is als de server dit moet doen voor 100 man. Natuurlijk heb ik wel gekeken naar een turn based system alleen er nooit in gekregen. Dus daar zullen performance issues komen op het moment van multiplier.

# Conclusie

Na het onderzoek komt er eigenlijk vrij duidelijke een paar dingen naar voren die moeten worden aangepast en gemaakt. Het eerste ding dat moeten worden gemaakt is dat de applicatie moet gaan werken op een Distributed database systeem. Waarom? Wel het is voordeliger voor de applicatie om dit te verdelen over het geval van services.

Dit werkt in het geval dat een services offline moet. Hierdoor heb je niet het probleem dat de hele applicatie eronder ligt en stil moet. Ook op moment van issues is dit een pluspunt. Verder komt hier door dat de applicatie uit elkaar moet worden getrokken. Wat ik hiermee bedoel is dat er een FE moet komen een BE Api die de services dan aanroept en beheert.

Om dit verder te supporten wil ik gebruiken gaan maken van O2Auth systeem. Voor nu lijkt google de beste optie omdat bekant iedere gebruiker wel een google account heeft. Ook laat ik de optie open voor een apart login/register form. Dit om nog steeds een optie te geven aan de gebruiker om in te kunnen loggen zonder een services van ergens anders.

Verder gaat de applicatie dan ook gebruiken maken van JWT tokens. Dit om de applicatie toch nog een beetje meer security te geven. Ook om te voorkomen dat de API straks wordt gehackt en dat er malicious files worden geüpload naar de API die dan weer worden gedownload door die clients.

### Prototyping ideas.

Het idee is om dit eerst op een Test base Driven development te gaan testen. Wat ik hier mee bedoeld is dat er als te volgende een prototype wordt gemaakt:

1. Er wordt een project opgezet voor een Login/Register op both FE en BE.
2. Op de BE wordt ook een optie toegevoegd om 1 locatie op te halen.
3. Voor deze 2 opties worden daarna ook testen geschreven en gemaakt op bepaalde punten te testen en zien of het goed gaat
   1. Denk aan opties als het login en register form
   2. Zien of de gebruiker een JWT token terug krijgt en kan gebruiken
   3. Te zien of je niet zonder in te loggen op de API kan
   4. Ook om te kijken of de API data terug geeft aan de FE in verband van locatie
4. Na dat dit gemaakt is en getest wordt de API verder los getrokken en er een services van gemaakt.
5. Nadat 4 is getest en wekt hoort de Google O2auth services te worden gekoppeld en getest. Het uiteindelijke doel na al dit is om een structuur die veilig is en eruit ziet als in Image 1.4 Verwachte Data structuur
6. Het uiteindelijke doel is om een applicatie te hebben die getest is op de security en ook werkt via services en hierna langzaam uit te breiden voor de rest van de applicatie.

# Bronnen

ACM Ethics. (2022, 8 juni). *Software Engineering Code - ACM Ethics*. ACM Ethics - The Official Site of the Association for Computing Machinery’s Committee on Professional Ethics. Geraadpleegd op 15 juni 2022, van <https://ethics.acm.org/code-of-ethics/software-engineering-code/>

*DBMS | Types of Databases - javatpoint*. (2021). Www.Javatpoint.Com. Geraadpleegd op 15 juni 2022, van <https://www.javatpoint.com/types-of-databases>

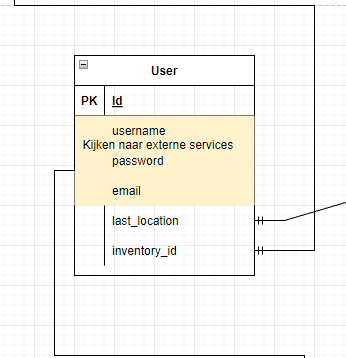
Dizdar, A. (2022, 21 april). *API Security: The Complete Guide to Threats, Methods & Tools*. Bright Security. Geraadpleegd op 16 juni 2022, van <https://brightsec.com/blog/api-security/>

Gaille, B. (2018, 25 augustus). *15 Centralized Database Advantages and Disadvantages*. BrandonGaille.Com. Geraadpleegd op 15 juni 2022, van <https://brandongaille.com/15-centralized-database-advantages-and-disadvantages/>

MacDonald, A. (2022, 28 april). *Cloud Storage vs. Local Storage: 19 Pros and Cons*. Carbide. Geraadpleegd op 15 juni 2022, van <https://carbidesecure.com/resources/cloud-storage-vs-local-storage-19-pros-and-cons/>

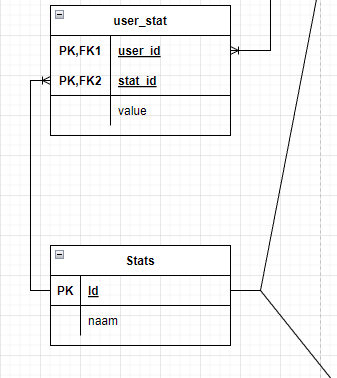
# Images

## Image 1.1 User table



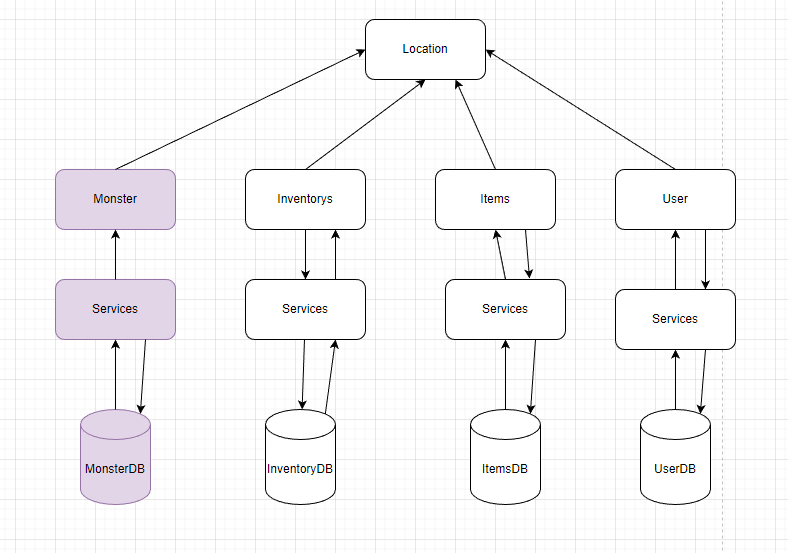
*Image 1.1 The user table structure*

## Image 1.2 Stats table



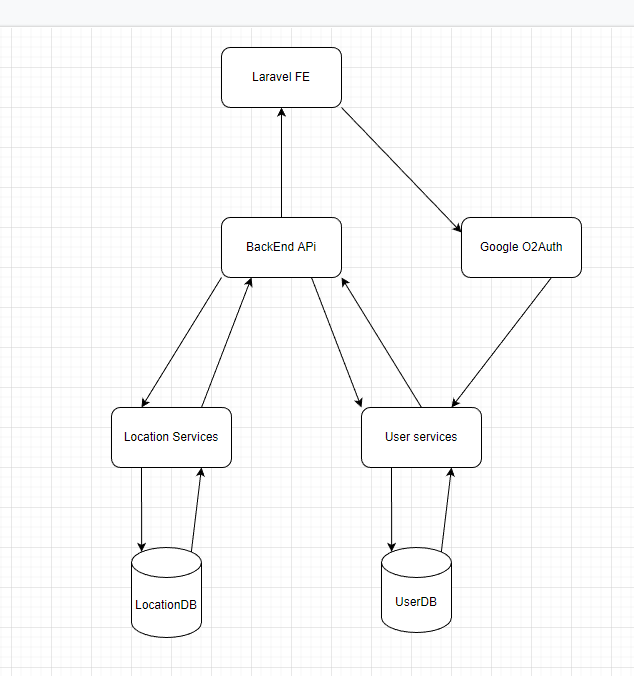
*Image 1.2 The stats table and a connection to the users table*

## Image 1.3 Structuur voor Distrubited database system



*Image 1.3 Structuur voor een database distrubted system*

## Image 1.4 Verwachte Data structuur



*Image 1.4 Data opzet naar alle stappen in de conclusie*