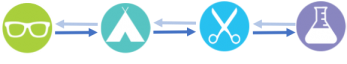
# Welke methodes van opslaan voor data kent Blazor?

## Doel

Het doel van dit onderzoek is het vinden van een geschikte manier voor het opslaan van data binnen de app. Waarbij er gedacht moet worden aan gebruiksvriendelijke en toekomstbestendige oplossing.

## Strategie

Zoals in het projectplan beschreven ga ik dit onderzoek doen volgens het “Choose fitting” pattern.



Choose fitting pattern

Dit betekend dat ik volgens de volgende volgorde ga werken:

1. Library: Literature study & community research

2. Field: Explore (user)requirements

3. Lab: Usability testing

4. Workshop: prototyping

## Uitwerking

Voor het uitvoeren van dit onderzoek heb ik eerst verschillende documentaties bestudeerd. Er zijn twee omgevingen waar data opgeslagen kan worden; server-side storage en de client-storage, in het geval van Blazor de webbrowser. Zoals ik al eerder in het projectplan aangegeven had, is het voor deze app van belang dat data opgeslagen kan worden op de client, zodat de inspecteur zonder internet verbinding gebruik kan blijven maken van de app. Hierbij wordt dus meteen de server-side storage uitgesloten, aangezien er dan een constante verbinding moet zijn, welke er in ons geval niet is.

Nu de keuze voor client-side storage is gemaakt gaan we kijken welke mogelijkheden er hier voor zijn. Het idee is om eerst een beeld te krijgen welke mogelijkheden welke functionaliteiten hebben. vervolgens wil ik van de gekozen methodes elk een demo maken, waarin er een bulk data van het bestaande backend wordt gedownload. Vervolgens wil ik die data kunnen manipuleren, waarna ik er een aantal tests mee uit ga voeren. Zoals de browser sluiten, nieuwe tabs openen naar dezelfde URL etc. Het idee daarachter is dat ik dan snel een beeld krijg of een gekozen methode effectief is voor hetgeen wat er in de toekomst mee gedaan gaat worden.

Er zijn twee standaard mogelijkheden volgens Microsoft.

**Methode 1, localStorage:**

LocalStorage is gebonden aan de browser zelf. Wanneer een gebruiker de pagina reload of opnieuw opend blijft de opgeslagen data en state bestaan. Wanneer een gebruiker een nieuw tabblad zou opnen, is diezelfde data ook vannuit dit nieuwe tabblad te gebruiken.

**Methode 2, sessionStorage:**

SessionStorage is gebonden aan de tab waar de gebruiker zich in bevind. Wanneer de gebruiker zijn pagina refreshed blijft de state bestaan. Wanneer de tab gesloten wordt is alles kwijt. Dit lijkt op voorhand al niet een geschikte kandidaat. Het is al bij voorbaad niet uit te sluiten dat een gebruiker altijd maar een tabblad geopend heeft, en al helemaal niet dat deze nooit gesloten wordt.

Meteen valt al op dat mehode 2, de sessionstorage, niet in aanmerking komt voor hetgeen wat we willen bereiken. Het is natuurlijk niet handig om maar van een methode afhankelijk te zijn, dus ik ga opzoek naar extra libraries die hier misschien wel in kunnen voorzien. Methode 1 ga ik in ieder geval wel verwerken in en proof-of-concept.

Ook methode 1 heeft meteen al de beperking dat er maar 5MB gebruikt kan worden. Dit is natuurlijk niet wenselijk, aangezien de metingen veelal groter kunnen worden. We moeten dus opzoek naar andere methodes die niet standaard geleverd worden.

Een andere manier die ik tegenkwam, en ik eerder heb gebruikt met Javascript applicaties, is IndexedDb. Natuurlijk kan je een IndexedDb ook gebruiken door midel van Javascript, maar omdat ik met dit onderzoek antwoord wil geven op de vraag of je deze applicatie met alleen Blazor kan maken, wil ik dus op zoek gaan naar een methode binnen Blazor. Wanneer ik zocht op de zoekterm “blazor indexeddb” kwamen een aantal packages naar voren, echter geen enkele was van Microsoft zelf. Normaliter gebruik ik liever zo min mogelijk packages van onafhankelijke developers, maar er was nu dus geen andere keus. Daarnaast heb ik gekeken in de NuGet library, waar diezelfde package (TG.Blazor.IndexedDB) bovenaan stond betreft downloads. Ik heb me dan ook hierin verdiept. Na het bekijken van een aantal tutorials, leek het mij een juiste keuze om deze te verwerken in een proof-of-concept.

Nu ik twee kandidaten had, begon ik aan het opzetten van een project in Visual Studio, te beginnen met de IndexedDb, aangezien mijn hoop daar min of meer op gevestigd is.

Ik maak een priject aan genaamd … en installeer meteen de juiste package door middel van “NuGet\Install-Package TG.Blazor.IndexedDB -Version 1.5.0-preview” in te voeren in de package manager console. Alles staat nu klaar voor mijn poc en ik kan een de slag.

Eerst zorg ik er voor dat de IndexDB toegevoegd wordt aan de service collection mijn app. Ik maak een nieuwe database met de naam “VSR-KMS” en maak een tabel aan, genaamd “Audit”. Ik maak een veld AuditNumber aan, wat autoincremented is. Dit doe ik omdat de audits al een Id hebben vanuit het backend, welke ik vervolgens wil gaan indexeren, zoals te zien op het screenshot. Zoals ook te zien is dat er nog geen data aan de meting hangt. Hiervoor heb ik bewust gekozen omdat ik Afbeelding met tekst

Automatisch gegenereerde beschrijvingeerst wil gaan uitzoeken voor het werkt, voordat ik het complexer wil gaan maken.

Ook maak ik een nieuwe folder aan, genaamd “Models”, waar ik alle modellen die ik wil gaan gebruiken in plaats. Hier maak ik een nieuw model aan “Audit.cs”.

Afbeelding met tekst, schermafbeelding, scherm

Automatisch gegenereerde beschrijving

Vervolgens moet ik de gebruikte dependency in de pagina waar ik ga werken injecten.

Afbeelding met tekst

Automatisch gegenereerde beschrijving

Eerst wil ik een knop gaan aanmaken die records aanmaak en vult met random mock data. Hiermee kan ik testen hoe makkelijk er records toegevoegd kunnen worden. Ook kan er dan getest worden hoe het verwijderen van records werkt.

Na het aandachtig doorlezen van de README op Github ben ik met het volgende gekomen. Ik heb twee knoppen gemaakt, AddItem om een record toe te voegen en GetAudits om te bekijken welke items er allemaal opgeslagen zijn in de database.

Met de method CreateString kan ik een random grote string genereren. Hiermee simuleer ik de Afbeelding met tekst

Automatisch gegenereerde beschrijvingmeting.

Nu probeer ik de applicatie te draaien, om vervolgens een meting toe te voegen. Helaas krijg ik een foutmelding, “Could not load type 'Microsoft.JSInterop.DotNetObjectRef' from assembly 'Microsoft.JSInterop, Version=3.0.0.0, Culture=neutral”.

Na veel zoekwerk en lezen op Github kom ik er achter dat dit te maken heeft met het niet ondersteunen van de laatste .Net versies, waaronder versie 6. Dit doet mij besluiten deze package zeker niet te gaan gebruiken. Ik baal er van, aangezien er niet veel andere veelbelovende IndexedDb opties zijn. Ik kom tot de conclusie dat ik misschien niet genoeg voorwerk heb gedaa, wat er tot heeft geleid dat ik een POC heb opgezet, waarvan het bij voorbaad al niet zou lukken.

Nu ik geleerd heb van niet genoeg voorwerk doen voor het kiezen van een package, ga ik nu anders te werk. Ik ga eerst veel lees en uitzoekwerk uitvoeren. Na veel uitzoeken en lezen op Github en Nuget kom ik tot de ontdekking dat er ene doorstart is geweest op Reshiru. Deze is omgeschreven en verbeterd, zodat deze nu ook in de nieuwste framework te gebruiken is. Meerdere mensen zie ik werken met dezelfde versies die ik zelf ook gebruik. Deze nieuwe package is te vinden onder de naam “IndexedDB.Blazor”. De manier van initialiseren van de package in het project is niet veel anders dan die van Reshiru. Wel is het velen malen makkelijker een datastore(database) te realiseren. Dit doe je door onderstaande.

Afbeelding met tekst

Automatisch gegenereerde beschrijving

Singleton instance toevoegen aan service

Afbeelding met tekst

Automatisch gegenereerde beschrijving

Het initialiseren van de database met Au

Het Audit model is hetzelfde als in de voorgaande POC. Het enige verschil is dat er nu aangegeven kan worden welke property gebruikt wordt als key door de IndexedDB.

Afbeelding met tekst

Automatisch gegenereerde beschrijving

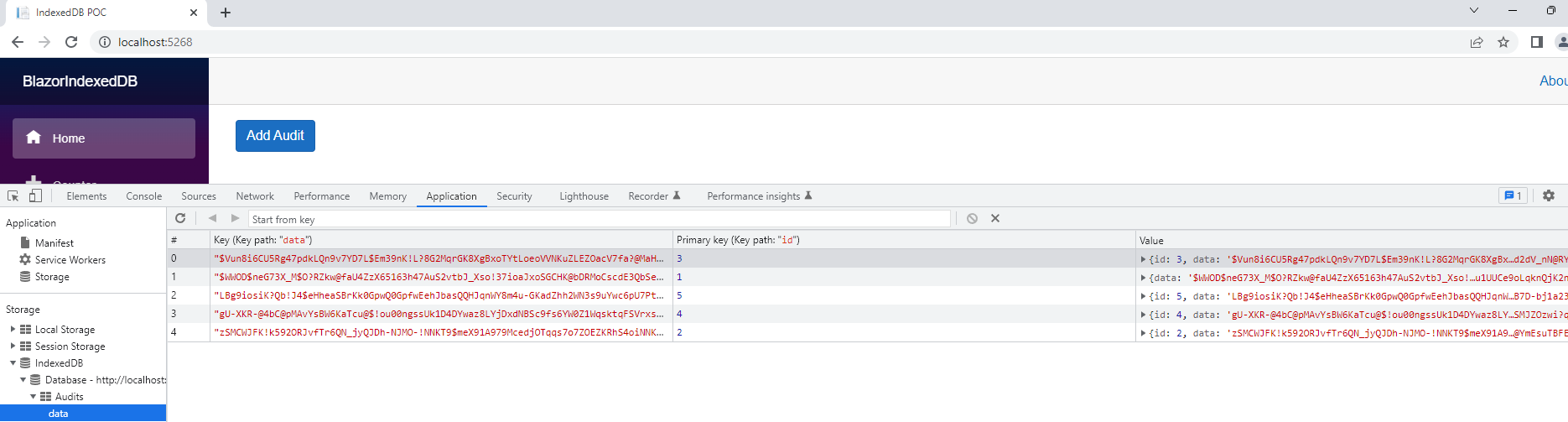
Om de package te testen gebruik ik, op een kleine aanpassing na, dezelfde index als voorheen.

Afbeelding met tekst

Automatisch gegenereerde beschrijving

## Toevoegen van een Audit

Het idee zou nou moeten zijn dat wanneer ik op de button “AddItem” klik er een record toegevoegd zal worden in de IndexDb van mijn browser. Ik ga mijn project nu runnen en kijken of het daadwerkelijk werkt. Zoals te zien is op regel 20 maak ik eerst gebruik van weinig data, aangezien het nog niet van belang is om een stresstest uit te voeren.

*F* *igure 4: Toevoegen van data in IndexdDb succesvol verlopen*

Na 5 maal op de “Add Audit” knop te hebben gedrukt en dit keer geen foutmeldingen te hebben gezien, klik ik op de knop refresh binnen het IndexedDb scherm. Ik zie dat het plan heeft gewerkt en dat er vijf ‘metingen’ zijn aangemaakt.

Voordat ik beginnen met het testen van de modify en delete, schrijf ik eerst een stukje code om alle data uit de database weer te geven.

*F*Afbeelding met tekst, scherm, elektronica

Automatisch gegenereerde beschrijving *igure 5: Code voor het weergeven van alle bestaande data*

## Aanpassen van een Audit

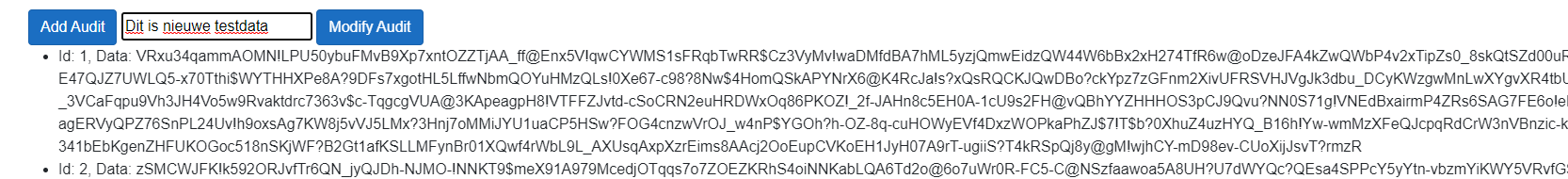
Nu ga ik kijken of ik ook makkelijk metingen kan opvragen en aanpassen. Ik ga een tekstveld en een knop aanmaken in de index. Wanneer ik op de knop “Modify Audit” klik wordt de Data van eerste meting in de IndexDb aangepast, naar de tekst in het tekstvak. Ik schrijf een stuk code wat de Data van eerste Audit in de database aanpast naar de opgegeven string.

Afbeelding met tekst

Automatisch gegenereerde beschrijving

Nieuwe code voor invoeren nieuwe data en

Hierna start ik de applicatie en voer ik de nieuwe data in het tekstvak in. Hierna druk ik op de knop “Modify Audit”, waarna ik de pagina ververs met F5.



Ik zie dat de data van de eerste meting inderdaad is aangepast, wat bevestigd dat ook het aanpassen werkt.

Afbeelding met tekst

Automatisch gegenereerde beschrijving

De aangepaste data

## Verwijderen van een Audit

Vervolgens wil ik testen of het verwijderen van Audits ook net zo makkelijk werkt. Ik maak een method voor het verwijderen van alle audits. Zo zou ik na het gebruiken van deze method, meteen weer een schone database hebben.

Afbeelding met tekst

Automatisch gegenereerde beschrijving

Aangemaakte knop en method voor het vewi

Opnieuw start ik de app en klik op de knop “Remove Audits”, waarna ik de pagina ververs met F5. Ik zie dat de lijst leeg blijft. Om dit te verifieren kijk ik handmatig in de database. Maar ook wanneer ik handmatig in de database kijk is alles leeg.

Afbeelding met tekst

Automatisch gegenereerde beschrijving

Zowel de lijst als de database zelf zijn

## Controleren overige requirements

Nu ik de drie hoofdeigenschappen van deze package succesvol heb getest, ga ik de overige belangrijke eisen testen.

## Werken met meerdere geopende tabbladen

Uit ervaring weten wij als makers van de app dat het lastig kan zijn het gebruik van de applicatie te controleren. Zo gebeurt het vaker dat inspecteurs verschillende tabbladen open hebben staan van dezelfde applicatie. Wanneer je bijvoorbeeld sessionstorage zou gebruiken, is de data van de applicatie niet te gebruiken tussen de verschillende tabs. Wanneer je het tabblad sluit, is die data ook weg. Dat zijn twee dingen die cruciaal zijn en ook meteen makkelijk zijn om te testen.

Aangezien ik al een POC heb waarmee makkelijk data aangemaakt kan worden ga ik dit testen door het gewoonweg te gaan proberen met de volgende stappen.

1. Ik open een tabblad (tabblad 1) en zorg dat de database gevuld is met een aantal “metingen”.

2. Ik open een nieuw tabblad (tabblad 2) met dezelfde url en pas daar de eerste record aan.

3. Ik ververs tabblad 1 en kijk of ik de aangepaste record zie.

Uitkomst: Ik zie de aangepaste record. Het is dus mogelijk om met verschillende tabbladen te werken.

## Verder werken na sluiten browser

Het komt natuurlijk wel eens voor dat een gebruiker de applicatie om wat voor een reden dan ook sluit. Het is dan natuurlijk niet gewenst dat de data van dat moment verdwijnt. Om te testen of de data ook nog bestaat na het sluiten van de browser ga ik de volgende stappen doorlopen.

1. Ik ‘Run’ de applicatie en maak wat records aan en ga wat data manipuleren.

2. Ik sluit de applicatie dmv het sluiten van de browser.

3. Ik start de applicatie opnieuw.

Uitkomst: Alle data die bestond voor het sluiten van de applicatie bestaat nog in de juiste state.

## Kan de IndexedDB ‘onze’ maximale payload aan data aan?

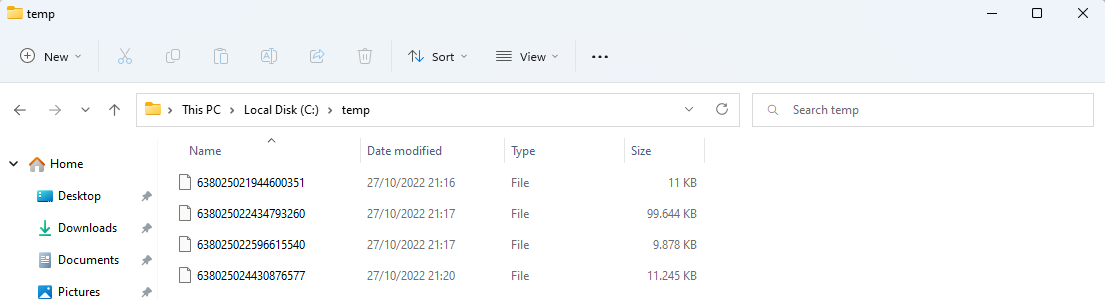
Zoals al eerder gezegd kunnen gebruikers meerdere metingen tegelijkertijd ophalen. Sommige metingen bevatten SVG tekeningen die soms tot maximaal 10m kunnen zijn. In het meest extreme geval zou een inspecteur 10 van die metingen per keer op willen halen. Ik ga nog een stap verder en ga kijken of ik 5 metingen van 10mb kan binnenhalen. Als dit lukt is deze package geslaagd voor deze stresstest.

1. Ik pas de code aan zodat er records van 10mb worden aangemaakt bij het drukken op de “Add Audit” knop.

*\*De JSON bestanden die van- en naar onze server worden gestuurd zijn maximaal 10mb. Ik heb een klein stukje code geschreven waarmee ik kan vaststellen hoeveel chars uit mijn eerdere code, gelijk staan aan een JSON bestand van 10MB. Met deze code maak ik bestand aan op mijn eigen filesystem tussen een range van 10.000 en 100.000 chars. Wanneer de file ongeveer 10MB groot is, weet ik wat het juiste aantal chars is. Met dat aantal chars ga ik de genoemde 15 records aanmaken.*

*F*Afbeelding met tekst

Automatisch gegenereerde beschrijving *igure 10: Code voor het bepalen van de grootte van de record*

***F*** ***igure 11: De aangemaakte files voor het bepalen van de juiste hoeveelheid chars***

*Met de range tussen de 10.000.000 en 12.000.000 chars was de file ongeveer 10MB groot. Nu weet ik dus dat dat aantal gebruikt gaat worden in deze test.*

2. Ik pas de weergave van de index pagina aan, zodat er dadelijk niet 5 keer een record van 10mb op het scherm wordt getoond. In plaats daarvan wordt er nu het id samen met de grootte van de record weergegeven.

3. Ik run de applicatie, maak de database schoon een maak 5 nieuwe records aan

4. Ik voor de tests “Werken met meerdere geopende tabbladen” en “Verder werken na sluiten browser” nogmaals uit.

Uitkomst: De data is aanpasbaar en dus is de test geslaagd.

## Conclusie

Ik ben erg tevreden over deze package, des te meer omdat de code hiervan is vrijgegeven volgens het opensource principe. Ik heb alle code van de package doorgespit en deze lijkt erg makkelijk te onderhouden en aan te passen. Ook alle test die ik heb uitgevoerd zijn met goede scores uitgevoerd. Er kan nu dus worden gezegd dat dit mijn gekozen manier is voor het opslaan van data binnen Blazor applicaties.

Ook ben ik mij er van bewust dat ik in de eerste instantie te snel met een methode in zee ben gegaan. Dit wilt zeggen dat ik in mijn toekomstige onderzoeken meer aandacht moet besteden aan het vooraf doen van onderzoek. Dit heeft mij uiteindelijk veel kostbare tijd gekost.

# Bronnen

<https://github.com/Jinjinov/IndexedDB.Blazor>

<https://www.nuget.org/packages/TG.Blazor.IndexedDB/1.5.0-preview>

<https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage>

<https://www.thomasclaudiushuber.com/2021/04/19/store-data-of-your-blazor-app-in-the-local-storage-and-in-the-session-storage/>

<https://learn.microsoft.com/en-us/aspnet/core/blazor/state-management?view=aspnetcore-6.0&pivots=server>

<https://blog.stevensanderson.com/2019/08/03/blazor-indexeddb/>

<https://www.nuget.org/packages?q=blazor+indexeddb>