**Advies**

Na een paar weken hadden we een paar developers geïnterviewd. We kregen de opdracht om daar ook een structuur in te vinden. Daardoor adviseerde we in elke interview een back-end en een frontend developer bij elkaar te zetten. Hierdoor kregen een soort discussie wat handig is. We wilde meerdere interviews houden, omdat we dan elke interview konden verbeteren. Zodat we meer informatie konden halen uit de volgende interview. Hierdoor kwamen er paar onderwerpen uit die we misschien konden automatiseren. We koosden voor Label translation, omdat ze hier soms een heel veel tijd kwijt zijn om het synchroon te houden. We probeerde een programma te bedenken waarop ze een hele hoop tijd kunnen besparen. Dit gaat meestal ook gepaard met frustratie. Het leek ons ook het interessantste omdat we meerdere algoritmes en gegevensverwerking kunnen inplanteren.

Ons advies is om WPF voor het programma. De reden hiervoor is dat je de programma lokaal op je computer kan gebruiken. Je hoeft niet meer aan de security te denken. De files worden nu ook steeds lokaal bewerkt. Dus de files kunnen dan op hetzelfde level worden bijgewerkt. Het programma kan ook op 1 harde schijf locatie worden gerund. We adviseerde wel om bij JSON te blijven, omdat je bijvoorbeeld bij een database veel meer performance vraagt. Ook omdat we een offlineprogramma maakt, wilde we geen online connectie maken. Stel dat het internet uitvalt, dat je dan gewoon kan doorwerken en als er iets verkeert gaat. Dan kan je het heel snel met git het weer terugzetten. WPF heeft al een goede ontwikkeling gehad. Zodat er al veel informatie beschikbaar is op het internet. Het is heel makkelijk als iets niet goed werkt van WPF, dan kan je of zelf wat schrijven of een nuget packets installeren. Dit zorgt ervoor dat het flexibel blijft. Een voorbeeld die we gaan gebruiken is Mahapps voor het design. Dit maakt WPF toch een stuk moderner.

Het programma wordt een .exe file, die je niet steeds hoeft te installeren. Dit zorgt ervoor dat het maar op één plek staan en iedereen kan het gebruiken. Zelfs kan je het overal verplaatsen en nog gebruiken. De savefile wordt opgeslagen in %appdata%. De savefile wordt een serializable object. Er wordt niet zo veel gegevens opgeslagen en alleen onze programma gaat er gebruik van maken. Op deze manier zorg er ook voor dat er geen corrupt data in voor. Als er nu iets veranderd, dan zal er een nieuwe file gemaakt moeten worden. Dit zorgt ervoor dat iedere lokale account toch zijn eigen ervaring krijgt. Je zou eventueel zelfs dit programma naar een klant kunnen sturen en hem alles laten invullen.

Ook hebben we geadviseerd over hoe de applicatie eruit komt te zien. We hadden eerst gepraat met een designer, zodat we goede argumenten hadden voor het design. Bijvoorbeeld waarom die ene item op die plek staat. We kwamen erachter dat een beetje dezelfde design moesten nemen als bijna elke andere Windows programma. Dit zorgt ervoor dat er al een vertrouwens gevoelens zijn. Zodat de gebruiker de gebruiker gelijk kan beginnen. Ook hebben we een manier van geadviseerd dat je heel snel labels achter elkaar kan uitvoeren. Er staat een balk onderaan de lijst waar je alles kan invullen en als je enter intoets. Dat je dan het toevoegt.

We moesten ook gaan nadenken of we moesten adviseren om te veranderen van vertalingen ophalen. Maar we kwamen al snel achter dat de front-end developers een fijne manier vinden van werken. Je kan dan heel makkelijk met talen als javascript de JSON-files uitlezen en dan kan je ermee werken. Zo kwamen we snel terug dat dit de goede manier is om de vertalingen op te halen.

Later in overleg met het bedrijf wilden we de structuur van de JSON-files veranderen. Nu moesten ze als er mark down was, dan parsen met een plusje in de naam zelf. Maar dat vonden we zonde, want elke keer als je zelf bedacht. Dan moest je dat op twee of meer plekken veranderen. In plaats van 1 labelnaam, splitsen we het nu in twee. Namelijk: value en isMarkdown. Nu kan je een boolean veranderen in true en false. Als je dan deze ophaalt, dan kan een website nu gelijk zien hoe hij het nu moet parsen. Ook hebben we dat alles in de JSON-file op alfabetische volgorde wordt gezet, omdat als je door de file lees. Dat je het handmatig sneller kan vinden.

Verder hebben we ook geadviseerd hoe we alle processen willen aanpakken. Dit wilde we graag in een scrum verband zien. Dan kunnen we elke sprint laten zien wat gedaan hadden. Door deze aanpak konden we bij ons zelf kijken of er knelpunten zijn. Door bijvoorbeeld requirements te maken kunnen we ook het overzicht houden waar we zijn en wat we nog moeten gaan doen. Aan het eind van elke sprint wilde graag een demo geven. We wilde graag oefenen om voor een paar mensen te presenteren.

Deze informatie hebben we steeds in een vergadering gepresenteerd. Aan de hand van de voorbeelden en argumenten van hierboven. We deden dit in de vergadering/demo zodat ze met zijn allen vragen kon beantwoorden. De bovenstaande argumenten werden positief ontvangen. Voor de rest deden we alle aanvragen volgens de normen van VI Company.